2016

ProLAN

ProLAN DbConnector 1.0 32 & 64 bit

Руководство разработчика

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

Оглавление

Введение

Компонент ProLAN DbConnector (далее **PLDbConnector**) предназначен для выполнения запросов к локальной базе данных ProLAN, содержащей результаты опросов клиентов по различным вопросам. Компонент позволяет, не разбираясь со структурой таблиц и правилами хранения и выборки данных, быстро и удобно выполнять запросы по интересующим параметрам и получать результаты выборки в виде коллекции объектов. Данные выборки могут быть использованы для импорта во внешние системы или создания отчетов «на лету».

Компонент PLDbConnector представляет собой 32-х или 64-х разрядный COM (ActiveX) компонент, который устанавливается на любой компьютер локальной сети, с которого возможен доступ к базе данных. Компонент позволяет из любого Windows приложения или службы, поддерживающей взаимодействие с COM, выполнять запросы на выборку данных.

Установка компонента

Компонент может быть установлен на компьютер с ОС Windows 7 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 PLDbConnector вы можете загрузить по ссылкам:
http://www.prolan.ru/files/freetools/DbConnectorSetup.exe
http://www.prolan.ru/files/freetools/DbConnector64Setup.exe

Скачайте дистрибутив и запустите на выполнение файл DbConnectorSetup.exe или DbConnector64Setup.exe (потребуются права администратора).

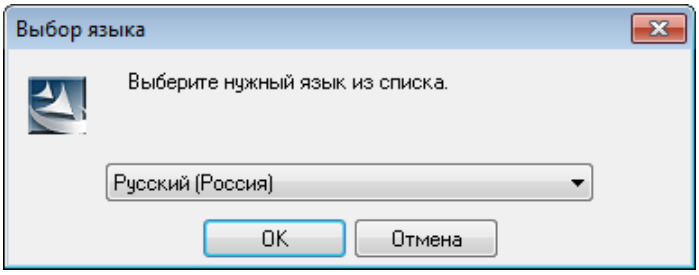

Рис.1. Выберите язык установки

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.

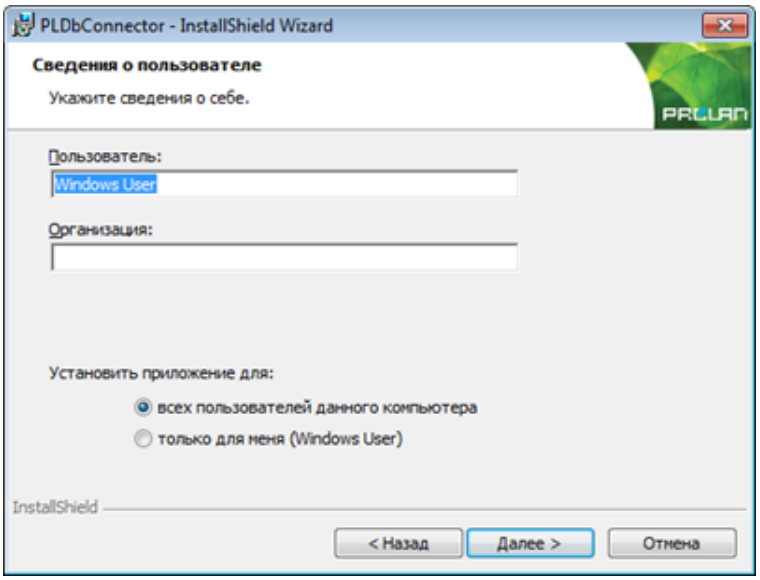

Рис.2. Сведения о пользователе

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 строки соединения с базой данных 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.

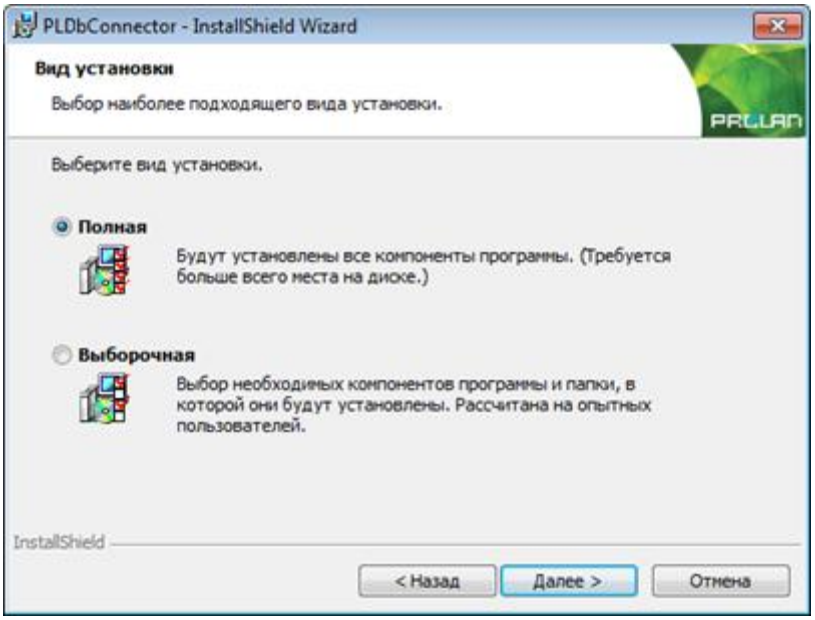

Рис.3. Вид установки

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.

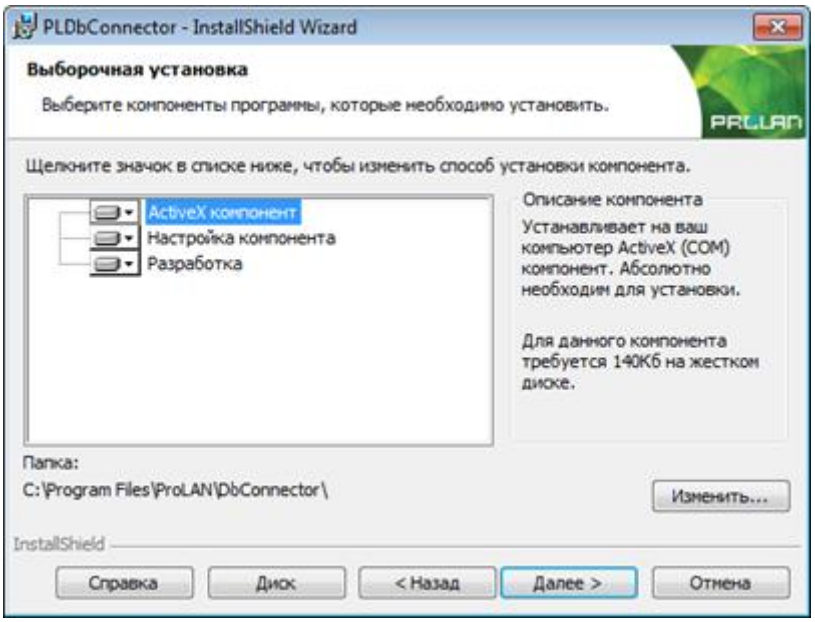

Рис.4. Выборочная установка

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

Внимание! Независимо от выбора папки установки, сам COM компонент DbConnector устанавливается и регистрируется в системе в папке C:\Program Files\Common Files\ProLAN\DbConnector либо C:\Program Files (x86)\Common Files\ProLAN\DbConnector 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 → DbConnector** или **DbConnector64** со значками запуска программ настройки COM компонента и на просмотра данного руководство разработчика.

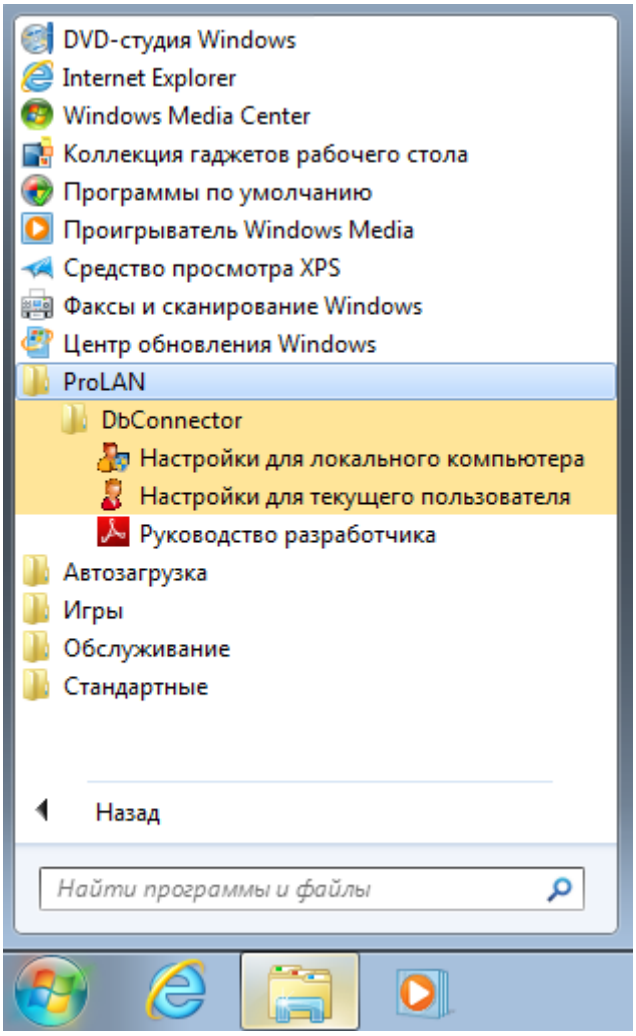

Рис.5. Пункты меню Window для запуска программ настройки компонента.

Настройки компонента

Компонент PLDbConnector и локальная база данных (MS SQL Server), как правило (но не обязательно), находятся на разных компьютерах локальной сети. Для выполнения запросов к базе данных компоненту необходимо сообщить **строку соединения**. Строка соединения может быть сообщена компоненту непосредственно в коде приложения, использующего компонент. Но передача значения строки соединения в коде приложения приводит к дополнительным затратам разработчика, хотя непосредственно к выполнению запросов отношения не имеют. Кроме того, строка соединения это условно постоянная величина. Поэтому, в большинстве случаев удобно выполнить предварительную настройку COM компонента и сохранить строку соединения 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 (строка соединения) 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 PLDbConnector 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.

Настройки компонента для текущего пользователя

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

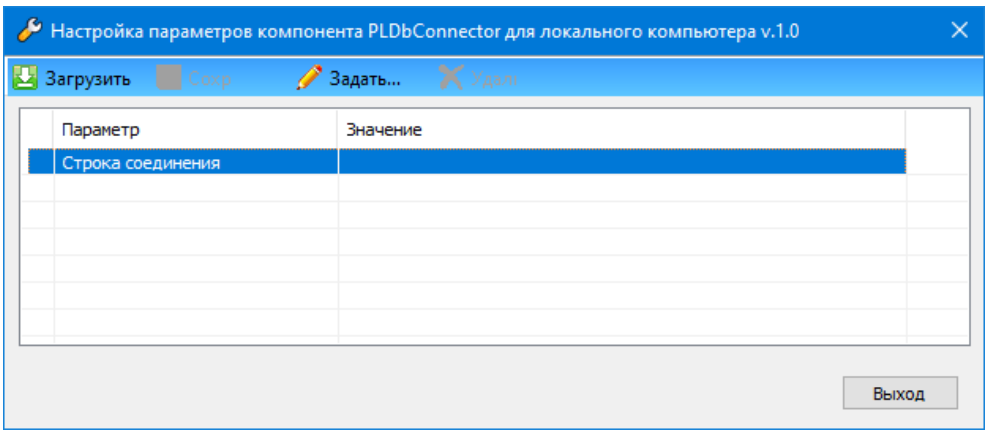

Рис.6. Окно диалога программы настроек для текущего пользователя.

Выберите в списке параметров **“Строка соединения с Базой данных”** и нажмите панели инструментов кнопку **“Задать”**. Задание и изменение строки соединения выполняется в стандартном Window диалоге **“Свойства передачи канала”**. На закладке **“Поставщик Данных”** выберите **“Microsoft OLE DB Provider for SQL Server”**.

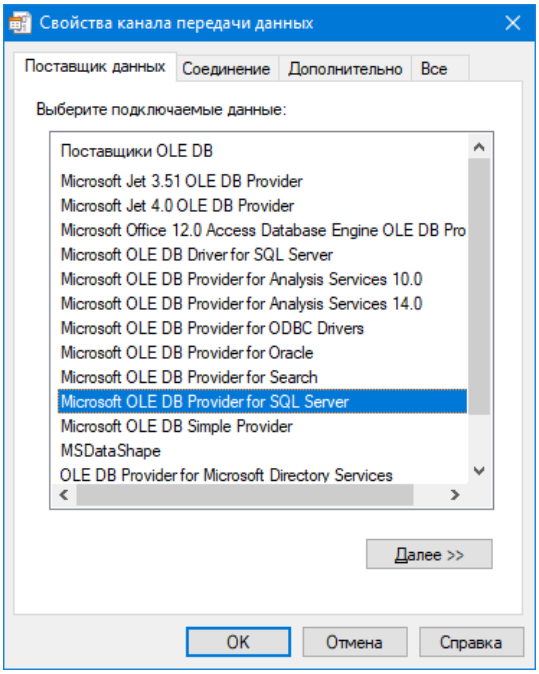

Рис.7. Строка соединения. Поставщик данных.

Перейдите на закладку **“Соединение”** (Рис.8). Выберите в выпадающем списке или задайте вручную имя или IP-адрес сервера MS SQL. Для бесплатных редакция MS SQL сервера (Express) после имени сервера задайте текст “\SQLEXPRESS”. Если сервер находится на этом компьютере, то вместо имени сервера можно указывать точку, например **“.\ SQLEXPRESS”**. Выберите вариант способа аутентификации пользователя на сервере MS SQL. Обычно задается аутентификация с использованием имени пользователя и пароля. В этом случае введи имя пользователя и пароль, и обязательно включите опцию **“Разрешить сохранение пароля”**.

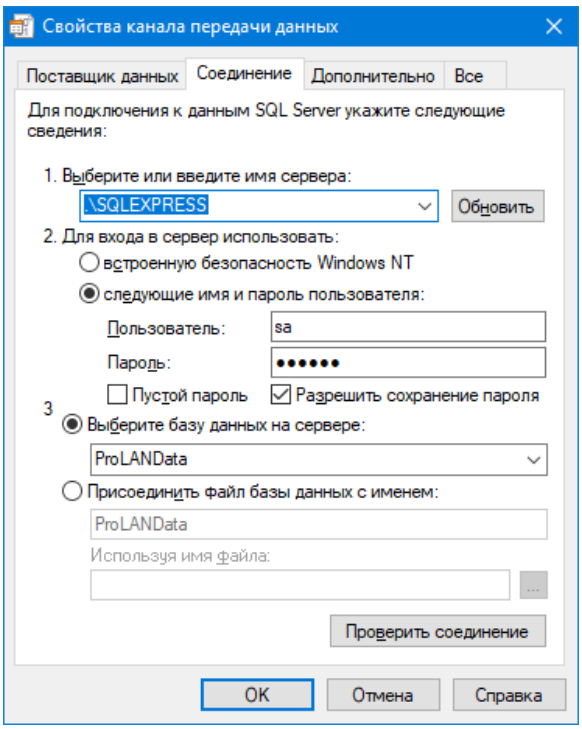

Рис.8. Строка соединения. Соединение.

Нажмите кнопку **“Проверить соединение”**. Если что-либо задано неверно, вы увидите сообщение с описанием ошибки. В противном случае будет выдано сообщение **“Проверка соединения выполнена”**. В выпадающем списке **Выберите базу данных на сервере**. Обычно это база данных **ProLANData**. Нажмите кнопку **“ОК”**. При возврате в главное окно программы значение сформированной строки соединения будет отображено.

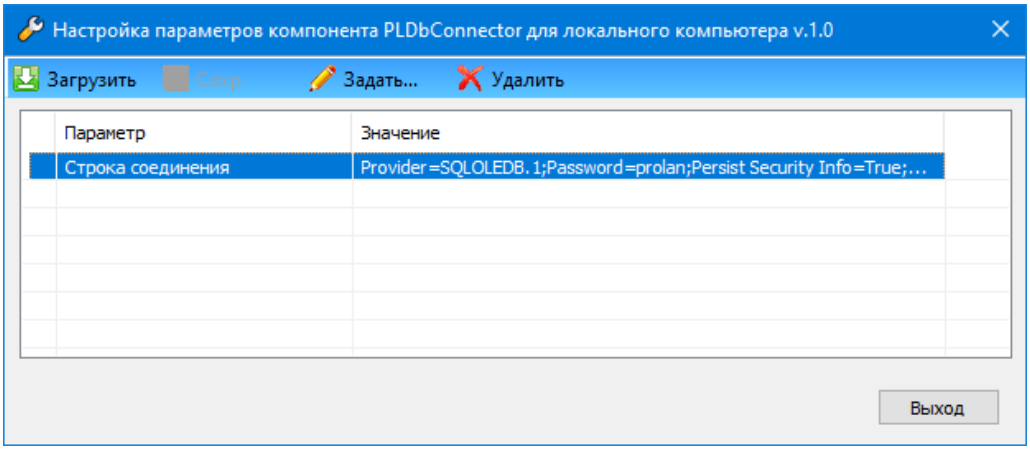

Рис.9. Строка соединения задана.

Нажмите кнопу “Сохранить” в панели инструментов для записи значения настроек в реестр.

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.

Объект DbRequest

Создание объекта DbRequest 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, например среда разработки 1С. 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++.

Основным объектом компонента DbConnector является объект класса **CDbRequest** с интерфейсом **IDbRequest**. С помощью него выполняются запросы к базе данных и выборка результата.

VBScript

В скриптовых языках объект создается с использованием строкового эквивалента идентификатора интерфейса (ProgID), имеющего значение `PLDbConnector.DbRequest` либо `PLDbConnector64.DbRequest` для 32-х и 64-х разрядной версии компонента:

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

'Переменная будет содержать объект
Dim l_objDbRequest
'Создаем объект для 32-х разрядной версии компонента.
Set l_objDbRequest = CreateObject("PLDbConnector.DbRequest")
'Для 64-х разрядной версии компонента создание объекта выглядит так
' Set l_objDbRequest = CreateObject("PLDbConnector64.DbRequest")

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

C++

Подключите в код программы файлы "PLDbConnector_i.c" и " PLDbConnector_i.h" (PLDbConnector64_i.c и PLDbConnector64_i.h для 64-х разрядной версии), которые вы можете найти в подкаталоге CPP папки установки. Подключите заголовочный файл PLDbConnectorExt.h с кодами ошибок и методиками компонента. Инициализируйте библиотеку OLE в коде потока.

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

```
#include "PLDbConnector_i.c"
#include "PLDbConnector_i.h"  // для 32-х разрадной версии
// либо
#include "PLDbConnector64_i.c"
#include "PLDbConnector64_i.h"      // для 64-х разрадной версии

#include "PLDbConnectorExt.h"

...

// SMART указатель на объект с интерфейсом IDbRequest
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
if(hr == S_OK) {
    // Объект создан
}
else {
    // Ошибка создания объекта. Описание ошибки можно извлечь из hr
}
// Удаляем объект. Не обязательно для SMART указателей
l_spIDbRequest = NULL;
...
```

Класс CDbRequest 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.

Выполнение выборки данных

Создав к коде приложения объект DbRequest:

- Задайте нужный вам для выборки набор свойств объекта:
  Строка соединение с базой данных, начальная и конечная даты выборки, другие свойства;
- Выполните выборку данных, используя один из методов:
  `DoSelect()` - синхронный запрос или
  `AsyncSelect()` - асинхронный запрос;
- В случае выполнения асинхронного запроса убедитесь, что выборка завершилась;
- Получите число выбранных записей и запросите у объекта выбранные данные.

Свойства объекта DbRequest

Объект поддерживает несколько свойств (Properties).

SVersion

Только для чтения. Возвращает строку, содержащую номер версии компонента. Для текущей версии компонента возвращается строка `"1.0"`.

VBScript

```
Dim l_sVersion
```

```
' Получаем номер версии компонента
l_sVersion = l_objDbRequest.SVersion
```

C++

```
HRESULT CDbRequest::get_SVersion([out, retval]BSTR* pValue);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrVersion;
hr = l_spIDbRequest->get_SVersion(&l_bstrVersion);
...
```

LastErrorCode

Код ошибки последней операции. Только чтение. Возвращает числовой код ошибки последней операции произведенной с объектом, к которым относятся задание некоторых свойств и вызов 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_OUT_OF_MEMORY	1	Система не смогла выделить память.
_LE_INVALIDARG	2	Неверный аргумент свойства/метода
_LE_UNKNOWN_ERROR	3	Неизвестная ошибка
_LE_CREATE_OBJECT_ERROR	4	Ошибка создания объекта
_LE_PROMPT_DB_CONN_STRING_ERROR	5	Ошибка создания строки соединения с DB
_LE_SELECTION_IN_PROGRESS	6	Запрос выполняется в данный момент
_LE_SELECTION_ERROR	7	Ошибка в процессе запроса или выборки данных
_LE_SELECTION_INTERRUPTED_BY_USER	8	Выполнение запроса было прервано пользователем

VBScript

```
Const _LE_NO_ERROR = 0
Const _LE_OUT_OF_MEMORY = 1
Const _LE_INVALIDARG = 2
Const _LE_UNKNOWN_ERROR = 3
Const _LE_CREATE_OBJECT_ERROR = 4
Const _LE_PROMPT_DB_CONN_STRING_ERROR = 5
Const _LE_SELECTION_IN_PROGRESS = 6
Const _LE_SELECTION_ERROR = 7
Const _LE_SELECTION_INTERRUPTED_BY_USER = 8

Dim l_nLastErrorCode
```

```
' Получаем код ошибки последней операции
l_nLastErrorCode = l_objDbRequest.LastErrorCode
```

C++

```
HRESULT CDbRequest::get_LastErrorCode(LONG* pErrorCode);
```

```
#include "PLDbConnectorExt.h"
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
LONG l_lLastErrorCode;
hr = l_spIDbRequest->get_LastErrorCode(&l_lLastErrorCode);
// В l_lLastErrorCode теперь находится код ошибки последней операции
If (lLastErrorCode == _LE_NO_ERROR) {
      // Нет ошибки
}
...
```

LastErrorText

Текст ошибки последней операции. Только чтение.

VBScript

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

C++

```
HRESULT CDbRequest::get_LastErrorText(BSTR* pErrorText);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrErrorText;
hr = l_spDbRequest->get_LastErrorText(&l_bstrErrorText);
// В l_bstrErrorText теперь находится описание ошибки последней опрации
...
```

ConnectionString

Получает или задает строку соединения с базой данных. Строка. Если были выполнены предварительные настройки компонента, то при создании объекта свойство ConnectionString автоматически получит значение, сохраненное в реестре компьютера на уровне локального пользователя (более высокий приоритет) или локального компьютера. В противном случае строка соединения будет пустой строкой. Строка соединения должна быть задана до выполнения запроса методом DoSelect() или AsyncSelect().

VBScript

```
Dim l_sOldConnectionString
' Получаем текущее значение
l_sOldConnectionString = l_objDbRequest.ConnectionString

If l_sOldConnectionString = "" then
  ' Задаем строку соединения
  l_objDbRequest.ConnectionString = "......"
EndIf
```

C++

```
HRESULT CDbRequest::get_ConnectionString(BSTR* pCurrentValue);

HRESULT CDbRequest::put_ConnectionString(BSTR newValue);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrConnectionString;
hr = l_spIDbRequest->get_ConnectionString(&l_bstrConnectionString);
// В l_bstrConnectionString теперь находится строка соединения
// Задаем другую строку соединения
hr = l_spIDbRequest->put_ConnectionString(CComBSTR(_T("......")));
...
```

BeginDate

Получает или задает дату и время начала диапазона выборки данных. Вещественное число двойной точности. В целой части числа содержится число дней от 30 декабря 1899 года. В дробной части содержаться часы, минуты и секунды как доли от суток. При создании объекта свойство получает значение даты/времени начала предыдущих суток.

VBScript

```
Dim l_dblBeginDate
' Получаем дату/время начала диапазона выборки данных
l_dblBeginDate = l_objDbRequest.BeginDate
' Задаем другую дату - час назад от текущего времени
l_objDbRequest.BeginDate = DateAdd("h", -1, now)
```

C++

```
HRESULT CDbRequest::get_BeginDate(DATE* pOlsValue);

HRESULT CDbRequest::put_BeginDate(DATE newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
```

```
DATE l_dblOldBeginDate;
hr = l_spIDbRequest->get_BeginDate(&l_dblOldBeginDate);
// Задаем другое значение – час назад от текущего времени

UDATE udateLocal;
udateLocal.wDayOfYear = 0;
GetLocalTime(&udateLocal.st);
DATE dblDate;
VarDateFromUdate(&udateLocal, 0, &dblDate);
dblDate -= ((double)1.0/24.0);

hr = l_spIDbRequest->put_BeginDate(dblDate);
if (hr != S_OK) {
      LONG l_lLastErrorCode;
      hr = l_spIDbRequest->get_LastErrorCode(&l_lLastErrorCode);
}
```

EndDate

Получает или задает дату и время конца диапазона выборки данных. Вещественное число двойной точности. При создании объекта свойство получает значение даты/времени начала текущих суток. Выборке данных производится для дат больших или равных начальной даты и меньших чем конечная дата диапазона выборки.

VBScript

```
Dim l_dblEndDate
' Получаем дату/время конца диапазона выборки данных
dblEndDate = l_objDbRequest.EndDate
' Задаем другую дату – час назад от текущего времени
l_objDbRequest.BeginDate = DateAdd("h", -1, now)
```

C++

```
HRESULT CDbRequest::get_EndDate(DATE* pOlsValue);

HRESULT CDbRequest::put_EndDate(DATE newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
DATE l_dblOldEndDate;
hr = l_spIDbRequest->get_EndDate(&l_dblOldEndDate);
// Задаем другое значение – час назад от текущего времени

UDATE udateLocal;
udateLocal.wDayOfYear = 0;
GetLocalTime(&udateLocal.st);
DATE dblDate;
VarDateFromUdate(&udateLocal, 0, &dblDate);
dblDate -= ((double)1.0/24.0);


hr = l_spIDbRequest->put_EndDate(dblDate);
...
```

BeginDateText

Получает или задает дату и время начала диапазона выборки данных в виде строки. Строка. Формат представлени “ГГГГ-ММ-ДД чч:мм:сс“. При задании значения свойства с неверным форматом даты/времени внутренний код ошибки объекта (свойство `LastErrorCode`) может принимать значения:

- _LE_INVALIDARG Неверный формат строки даты/ Дата задана неверно
- _LE_UNKNOWN_ERROR Ошибка при преобразовании в тип DATA

VBScript

```
Dim l_sBeginDateText
' Получаем дату/время начала диапазона выборки данных
l_sBeginDateText = l_objDbRequest.BeginDateText
' Задаем другую дату
l_objDbRequest.BeginDateText = "2021-07-10 15:00:00"
If l_objDbRequest.LastErrorCode <> _LE_NO_ERROR then
    'Дата задана невенно
    ...
End If
```

C++

```
HRESULT CDbRequest::get_BeginDateText(BSTR* pOldValue);

HRESULT CDbRequest::put_BeginDateText(BSTR newValue);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrBeginDateText;
hr = l_spIDbRequest->get_BeginDateText(&l_bstrBeginDateText);
// Задаем другую дату
hr = l_spIDbRequest->put_BeginDateText(CComBSTR(_T("2021-07-10 15:00:00")));
if (hr != S_OK) {
    // Дата задана невенно
}
...
```

EndDateText

Получает или задает дату и время конца диапазона выборки данных в виде строки. Строка. Формат представлени “ГГГГ-ММ-ДД чч-мм-сс“. При задании значения свойства с неверным форматом даты/времени внутренний код ошибки объекта (свойство `LastErrorCode`) будет олично от значения `_LE_NO_ERROR`.

VBScript

```
Dim l_sEndDateText
```

```
' Получаем дату/время конца диапазона выборки данных
l_sEndDateText = l_objDbRequest.EndDateText
' Задаем другую дату
l_objDbRequest.EndDateText = "2021-07-10 16:00:00"
```

C++

```
HRESULT CDbRequest::get_EndDateText(BSTR* pOldValue);

HRESULT CDbRequest::put_EndDateText(BSTR newValue);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_Request);
...
CComBSTR l_bstrEndDateText;
hr = l_spIDbRequest->get_EndDateText(&l_bstrEndDateText);
// Задаем другую дату
hr = l_spIDbRequest->put_EndDateText(CComBSTR(_T("2021-07-10 16:00:00")));
...
```

Technique

Получает или задает **методику**, по которой в базу данных были занесены данные. При записи в базу данных Методика может иметь следующие значения:

- _TECHNIQUE_LOYALTY_BUTTON – Данные сзязаны с оценкой по методике клиентсткой лояльности. Оценки деляться на положительные, отрицательные и нейтральные;
- _TECHNIQUE_TIME_WINDOW – Данные полученные по методике «Открытого окна». Методика предполагает, что оценки клиента производятся после сигнала нажатия кнопки продавца в течение заданного времени (окна);
- _TECHNIQUE_POLLS – Данные полученные в результате управляемых или стихийных опросов клиентов по различным вопросам;
- _TECHNIQUE_INITIATIVE – Данные, записанные в базу данных по произвольным (неформализованным) методикам

В чиловом эквиваленте Методика может иметь слкдующие значения:

Методика	Числовое значение	Пояснение
_TECHNIQUE_ALL	0	Значение может быть задано для выборки данных по любой методике
_TECHNIQUE_LOYALTY_BUTTON	1	Выборка данных, полученных по методике оценки клиентской лояльности
_TECHNIQUE_TIME_WINDOW	2	Выборка данных, полученных по методике «Открытого окна»
_TECHNIQUE_POLLS	3	Выборка данных, полученных в результате опросов клиентов
_TECHNIQUE_INITIATIVE	4	Выборка данных, полученных с использованием неформализованных методик

При создании объекта **DbRequest** свойство **Technique** получает значение **_TECHNIQUE_ALL**. Т.е. если значение свойства до начала выборки не будет изменено, то будут выбираться данные по любой методике. Если выборку необходимо производить по какой-либо конкретной методике, то

измените значение свойства перед началом выборки. При попытке задания некорректного значения, знчение свойства не изменяется.

VBScript

```
Const _TECHNIQUE_ALL = 0
Const _TECHNIQUE_LOYALTY_BUTTON = 1
Const _TECHNIQUE_TIME_WINDOW = 2
Const _TECHNIQUE_POLLS = 3
Const _TECHNIQUE_INITIATIVE = 4

Dim l_nTechnique
' Получаем текущее значение
l_nTechnique = l_objDbRequest.Technique
' Задаем другое значение
l_objDbRequest.Technique = _TECHNIQUE_LOYALTY_BUTTON
```

C++

```
HRESULT CDbRequest::get_Technique(LONG* pOldValue);

HRESULT CDbRequest::put_Technique(LONG newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
LONG l_nTechnique;
hr = l_spIDbRequest->get_Technique(&l_nTechnique);
// Задаем другое значение
hr = l_spIDbRequest->put_Technique(_TECHNIQUE_LOYALTY_BUTTON);
...
```

Location

Получает или задает Концентратор Данных – компьютер от которого информация об ответах клиентов на вопросы передается в базу данных. Строка с максимальной длиной в 255 символов. При создании объекта свойство получает значение пустой строки. Если при выполнении выборки значение будет содержать пустую строку, то имя Концентратора Данных не будет учитываться в условии выборки. В противном случае, выбираются данные, принадлежащие заданному концентатору.

VBScript

```
Dim l_sLocation
' Получаем текущее значение
l_sLocation = l_objDbRequest.Location
' Задаем другое значение
l_objDbRequest.Location = "legolas.office.prolan.ru"
```

C++

```
HRESULT CDbRequest::get_Location(BSTR* pOldValue);

HRESULT CDbRequest::put_Location(BSTR newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrLocation;
hr = l_spIDbRequest->get_Location(&l_bstrLocation);
// Задаем другое значение
hr = l_spIDbRequest->put_Location(CComBSTR(_T("legolas.office.prolan.ru")));
...
```

Panel

Получает или задает имя пульта на котором была нажата кнопка ответа. Имя пульта может быть обезличенным, например “Окно 1” или иметь имя сотрудника или клиента, например “Иванов П.М.”. Строка с максимальной длиной в 255 символов.

При создании объекта свойство получает значение пустой строки. Если при выполнении выборки значение будет содержать пустую строку, то имя пульта не будет учитываться в условии выборки. В противном случае, выбираются данные, принадлежащие заданному пульту.

VBScript

```
Dim l_sPanel
' Получаем текущее значение
l_sPanel = l_objDbRequest.Panel
' Задаем другое значение
l_objDbRequest.Panel = "Окно 1"
```

C++

```
HRESULT CDbRequest::get_Panel(BSTR* pOldValue);

HRESULT CDbRequest::put_Panel(BSTR newValue);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrPanel;
hr = l_spIDbRequest->get_Panel(&l_bstrPanel);
// Задаем другое значение
hr = l_spIDbRequest->put_Panel(CComBSTR(_T("Иванов П.М.")));
...
```

Question

Получает или задает текст вопроса, на который отвечал клиент. Строка с максимальной длиной в 255 символов. При создании объекта свойство получает значение пустой строки. Если при выполнении выборки значение будет содержать пустую строку, то текст вопроса не будет учитываться в условии выборки. В противном случае, выбираются данные, когда клиент отвечал на заданный вопрос. **Следует учитывать**, что текст вопроса присутствует только в данных, полученных по методике опросов (_TECHNIQUE_POLLS). Таким образом значение свойства, отличное от пустой строки имеет смысл задавать для выборки данных по методике _TECHNIQUE_POLLS.

VBScript

```
Dim l_sQuestion
' Получаем текущее значение
l_sQuestion = l_objDbRequest.Question
' Задаем другое значение
l_objDbRequest.Question = "Оцените качество обслуживания"
l_objDbRequest. Technique = _TECHNIQUE_POLLS
```

C++

```
HRESULT CDbRequest::get_Question(BSTR* pOldValue);

HRESULT CDbRequest::put_Question(BSTR newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrQuestion;
hr = l_spIDbRequest->get_Question(&l_bstrQuestion);
// Задаем другое значение
l_spIDbRequest->put_Question(CComBSTR(_T("Оцените качество обслуживания")));
hr = l_spIDbRequest->put_Technique(_TECHNIQUE_POLLS);
...
```

AnswerAlias

Получает или задает текст псевдонима варианта ответа вопроса, который дал клиент. Строка с максимальной длиной в 255 символов. Псевдомим ответа и текст ответа это разные вещи и могу отличаться. Псевдоним варианта ответа, как правило, короткий абстрактный текст, например “1”, который не меняется при изменении текста варианта ответа. При создании объекта свойство получает значение пустой строки. Если при выполнении выборки значение будет содержать пустую строку, то певдоним варианта ответа не будет учитываться в условии выборки. В противном случае, выбираются данные, когда клиент отвечал с заданным псевдонимом варианта ответа. Если в выборке задается псевдонима варианта ответа, то логично также задавать и текст вопроса (для методики _TECHNIQUE_POLLS). В противном случае, если в данных приутствуют ответы на разные вопросы, но с одинаковыми псевдонимами вариантов ответов, то будут выбраны все такие результаты.

VBScript

```
l_objDbRequest.Question = "Оцените качество обслуживания"
Dim l_sAnswerAlias
' Получаем текущее значение
l_sAnswerAlias = l_objDbRequest.AnswerAlias
' Задаем другое значение
l_objDbRequest.AnswerAlias = "1"    'Положительная оценка
```

C++

```
HRESULT CDbRequest::get_AnswerAlias(BSTR* pOldValue);

HRESULT CDbRequest::put_AnswerAlias(BSTR newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
l_spIDbRequest->put_Question(CComBSTR(_T("Оцените качество обслуживания")));
CComBSTR l_bstrAnswerAlias;
l_spIDbRequest->get_AnswerAlias(&l_bstrAnswerAlias);
// Задаем другое значение
l_spIDbRequest->put_AnswerAlias(CComBSTR(_T("1"))); // Положительная оценка
...
```

AttrName

Получаемые в результате выборки записи могут иметь (не обязательно) набор **Атрибутов**. Атрибут это пара строк произвольного назначения имеющая имя атрибута и значение атрибута. Например, `"Client_ID" : "Иванов П.С."` или `"Категория" : "Постоянный клиент"`.

Если необхдимо выбирать из базы данных записи с учетом имеющихся имен и/или значений атрибутов, то перед началом выборки их необходимо задать.

Используя свойство AttrName вы можете задаеть имя атрибута включаемое в условие выборки. Строка с максимальной длиной в 255 символов. Если при выполнении выборки значение будет содержать пустую строку, то имя атрибута не будет учитываться в условии выборки. В противном случае, выбираются данные, имеющие атрибуты с заданным именем.

VBScript

```
Dim l_sAttrName
' Получаем текущее значение
l_sAttrName = l_objDbRequest.AttrName
' Задаем другое значение
l_objDbRequest.AttrName = "Категория"
 'Выбирать записи, имеющие атрибут с этим именем
```

C++

```
HRESULT CDbRequest::get_AttrName(BSTR* pOldValue);

HRESULT CDbRequest::put_AttrName(BSTR newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrAttrName;
hr = l_spIDbRequest->get_AttrName(&l_bstrAttrName);
// Задаем другое значение
hr = l_spIDbRequest->put_AttrName(CComBSTR(_T("Категория")));
...
```

AttrValue

Используя свойство AttrValue вы можете задаеть значение атрибута включаемое в условие выборки. Строка с максимальной длиной в 255 символов. Если при выполнении выборки значение будет содержать пустую строку, то значение атрибута не будет учитываться в условии выборки. В противном случае, выбираются данные, имеющие атрибуты с заданным значением. Если необхдимо выбирать из базы данных записи с учетом имеющихся имен и/или значений атрибутов, то перед началом выборки их необходимо задать.

VBScript

```
Dim l_sAttrValue
' Получаем текущее значение
l_sAttrValue = l_objDbRequest.AttrValue
' Задаем имя и значение атрибута
l_objDbRequest.AttrName = "Категория"
l_objDbRequest.AttrValue = "Новый клиент"
```

C++

```
HRESULT CDbRequest::get_AttrValue(BSTR* pOldValue);

HRESULT CDbRequest::put_AttrValue(BSTR newValue);
```

```
...
CComPtr<IDbRequest > l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
CComBSTR l_bstrAttrValue;
hr = l_spIDbRequest->get_AttrValue(&l_bstrAttrValue);
// Задаем имя и значение атрибута
hr = l_spIDbRequest->put_AttrName(CComBSTR(_T("Категория")));
hr = l_spIDbRequest->put_AttrValue(CComBSTR(_T("Новый клиент")));
...
```

AsyncSelectInProgress

Только для чтения. Логическая величина. Возвращает TRUE, если процесс выборки (асинхронной или синхронной) в данный момент еще не завершен. В противном случае возвращает FALSE. Подробно использование метода описано в методе AsyncSelect().

VBScript

```
...
If l_objDbRequest.AsyncSelect Then
  While l_objDbRequest. AsyncSelectInProgress
    'Какие либо действия и проверки
  Wend
End If
...
```

C++

```
HRESULT CDbRequest::get_AsyncSelectInProgress(VARIANT_BOOL* pbProgress);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
VARIANT_BOOL* pbProgress;
l_spIDbRequest->AsyncSelect(&pbProgress);
If (pbProgress) {
    While (pbProgress) {
        Sleep(100);
        l_spIDbRequest->AsyncSelectInProgress(pbProgress);
    }
    ...
}
...
```

ResultRecordCount

Только для чтения. Число. Возвращает количество объектов (записей) выбранных из базы данных в результате выполнения запроса на выборку (методом **DoSelect** или **AsyncSelect**). Если возвращается значение -1, то это означает, что процесс выборки еще не закончен.

VBScript

```
' Выполняем запрос
If l_objDbRequest.DoSelect Then
  ' Получаем число выбранных объектов
  Dim l_nRecordCont
  l_nRecordCont = l_objDbRequest.ResultRecondCount
  ' В цикле запрашиваем результаты выборки по индексу
  ...
End If
```

C++

```
HRESULT CDbRequest::get_ResultRecordCount([out, retval] LONG* pValue);
```

```
...
HRESULT hr = l_spIDbRequest.DoSelect();
if (hr == S_OK) {
  LONG lResCount;
  l_spIDbRequest->get_ResultRecordCount(&lResCount);
  // В цикле запрашиваем результаты выборки по индексу
  ...
}
...
```

Методы объекта DbRequest

Методы (Methods) объекта можно разделить на следующие категории:

- Методы очистки ошибки последней операции **ClearLastError** и сброса свойств объекта и результатов выборки **Reset**;
- Метод задания строки соединения в стандартном графическом интерфейсе Windows **PromptConnectionString**;
- Метод, выполняющий синхронную выборку **DoSelect**;
- Методы связанные с асинхронной выборкой данных. Методы **SetCallbackNull**, **SetCallbackWindow**, **SetCallbackEvent** и **WaitForCallbackEvent** Задают один из механизмов обратного вызова при окончании асинхронной выборки;
- Метод запуска асинхронной выборки **AsyncSelect** и метод **StopAsyncSelect** для прерывания этого процесса;
- Метод, возвращающий объект **DbResultRecord** из коллекции выбранных записей

ClearLastError

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

VBScript

```
l_objDbRequest.ClearLastError
```

C++

```
HRESULT CDbRequest::ClearLastError(void);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
```

```
l_spIDbRequest->ClearLastError();
...
```

Reset

Сбрасывает значения свойств объекта в начальное состояние (как при создании объекта):

- Свойство **ConnectionString не изменят своего значения**;
- **Дата начала интервала выборки** устанавливается в дату/время начала предыдущих суток;
- **Дата конца интервала выборки** устанавливается в дату/время начала текущих суток;
- Свойство **Technique** устанавливается в значение _TECHNIQUE_ALL
- Свойства Location, Panel, Question, AnswerAlias, AttrName и AttrtValue устанавливаются в пустую строку

Не имеет параметров и не возвращает никаких значений.
Сброс свойств подготавливает объект к повторному использованию.

VBScript

```
l_objDbRequest.BeginDateText = "2021-07-10 15:00:00"
l_objDbRequest.EndDateText = "2021-07-10 16:00:00"
l_objDbRequest.Location = "legolas.office.prolan.ru"
l_objDbRequest.Technique = _TECHNIQUE_POLLS
l_objDbRequest.Panel = "Окно 1"
l_objDbRequest.Question = "Оцените качество обслуживания"
l_objDbRequest.AnswerAlias = "1"

...

' Выполняем запрос
l_objDbRequest.DoSelect
...
' Получаем результат выборки
...

' Сбрасываем значения свойств
l_objDbRequest.Reset
...
' Повторно используем объект
```

C++

```
HRESULT CDbRequest::Reset(void);
```

```
...
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
...
l_spIDbRequest.BeginDateText(CComBSTR(_T("2021-07-10 15:00:00")));
l_spIDbRequest.EndDateText(CComBSTR(_T("2021-07-10 16:00:00")));
l_spIDbRequest->put_Location(CComBSTR(_T("legolas.office.prolan.ru")));
l_spIDbRequest->put_Technique(_TECHNIQUE_POLLS);
l_spIDbRequest->put_Panel(CComBSTR(_T("Окно 1")));
l_spIDbRequest->put_Question(CComBSTR(_T("Оцените качество обслуживания")));
l_spIDbRequest->put_AnswerAlias(CComBSTR(_T("1")));
```

```
...
// Выполняем запрос
l_spIDbRequest->DoSelect()
...
// Получаем результат выборки
...

// Сбрасываем значения свойств
l_spIDbRequest->Reset();
...
// Повторно используем объект
```

PromptConnectionString

Формирование строки соединения с базой данных в стандартном диалоге Windows «Свойства канала передачи данных».

```
HRESULT CDbRequest::PromptConnectionString([in] BSTR bstrInVal,
                                           [out, retval] BSTR* pbstrOutVal);
```

Параметры:

bstrInVal

Строка, инициализационная начальным значением строки соединения. Заданное значение инициализирует поля диалога «Свойства канала передачи данных».

pbstrOutVal

Выходной параметр. Указатель на строку, которая будет заполнена значением строки соединения, сформированным пользователем в диалоге, если диалог закрывается по кнопке «OK». Если пользователь нажимает в диалоге кнопку «Отмена», то будет занесена пустая строка.

Примечание Метод может быть вызван из кода приложения, но не службы (Service), т.к. службы не могут иметь графический интерфейс.

VBScript

```
'Создаем объект
Set l_objDbRequest = CreateObject("PLDbConnector.DbRequest.1")
'Получаем текщую строку соединения из объекта
Dim strOldConnectionString
strOldConnectionString = l_objDbRequest.ConnectionString

Dim strNewConnectionString
strNewConnectionString = _
     l_objDbRequest.PromptConnectionString(strOldConnectionString)

'Если строка соединения в диалоге сформирована и нажата кнопка OK
If Len(strNewConnectionString) > 0 Then
  'Задаем для объекта эту строку соединения
  l_objDbRequest.ConnectionString = strNewConnectionString
End If
```

C++

```
...

CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);
CComBSTR l_bstrOldConnectionString;
hr = l_spIDbRequest->get_ConnectionString(&l_bstrOldConnectionString);
// В l_bstrConnectionString теперь находится строка соединения
// взятая из настроек
CComBSTR l_bstrNewConnectionString;
hr = l_spIDbRequest->PromptConnectionString(l_bstrOldConnectionString,
                                            l_bstrNewConnectionString);
if ((hr == S_OK) && lstrlen(l_bstrNewConnectionString)) {
    // Задаем другую строку соединения
    hr = l_spIDbRequest->put_ConnectionString(l_bstrNewConnectionString);
}
...
```

DoSelect

Выполняет синхронный запрос к базе данных с использованием заданной строки соединения и параметров выборки:

- Интервала дат/времен выборки;
- Методики (**Technique**);
- Имени Концентратор Данных (**Location**), если задано;
- Имени пульта (**Panel**), если задано;
- Текста вопроса (**Question**), если задан;
- Псевдонима варианта ответа (**AnswerAlias**), если задан;
- Имени (**AttrName**) и/или значения (**AttrValue**) атрибута, если заданы.

Перед выполненим метода, код ошибки последней операции с объектом автоматически устанавливается в _LE_NO_ERROR (0). Метод синхронный, т.е. не возвращает управление вызывающему коду до окончания работы кода метода. Выполнение может быть прервано за счет вызова метода StopAsyncSelect. Но вызвать метод StopAsyncSelect можно только в параллельном потоке кода. Такой механизм не является стандартным для синхронной выборки.

Метод производит следующие действия:

- Подключается к базе данных MS SQL, используя строку соединения;
- Формирует SQL запрос к базе данных на основе параметров выборки;
- Выполняет SQL запрос и получает результат выборки;
- Сохраняет данные выборки во внутренних объектах;
- Закрывает соединение с базой данных;
- Возвращает управление вызывающей программе и сообщает информацию об успехе или ошибке в процессе выполнения.

Метод возвращает TRUE, если выборка данных произведена успешно и FALSE в других случаях. Если при выполнении действий на любом этапе происход ошибка, то выполнение прерывается, код и описание ошибки заносится в значения свойст **LastErrorCode** и **LastErrorText**.

Возможны следующие коды ошибок при выполнении метода:

- _LE_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется". Ранее был запущен процесс асинхронной выборки (метод AsyncSelect), который на данный момент еще не завершен.
- _LE_SELECTION_ERROR – Ошибка в процессе соединения с БД или выборки данных. Используйте свойство **LastErrorText** для получения детального описания причины ошибки.
- _LE_SELECTION_INTERRUPTED_BY_USER –"Процесс выборки прерван пользователем". В процессе выборки был вызван метод StopAsyncSelect.
- _LE_UNKNOWN_ERROR – "Неизвестная ошибка в процессе выборки".

Примечание Использование синхронного метода **DoSelect** выборки данных оправдано в том случае, если вызывающий метод код (поток) может выделить некоторое время для выполнения процедуры запроса и выборки данных. В зависимости от параметров выборки и массива данных, процесс выборки может происходить практически мгновенно, но может и затянуться на несколько секунд. Если для программы это не критично (программа может подождать, либо запрос выполняется в отдельном потоке), то используйте метод **DoSelect**. В противном случае выполняйте запрос асинхронно (**AsyncSelect**).

VBScript

```
Dim l_objDbRequest
'Создаем объект
Set l_objDbRequest = CreateObject("PLDbConnector.DbRequest")

'Задаем значения для параметров для выборки
...

'Выполняем запрос синхронно
Dim l_Success
l_Success = l_objDbRequest.DoSelect

If l_Success Then
  'Получаем результат выборки
  ...
Else
  MsgBox "Ошибка выполнения запроса (" & l_objDbRequest.LastErrorCode & _
        "): " & l_objDbRequest.LastErrorText, vbOKOnly + vbCritical, "Ошибка"
  ...
End If
```

C++

```
HRESULT CDbRequest::DoSelect([out, retval] BOOL* pSuccess);
```

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. Если методу передается выходной параметр `pSuccess`, то в переменную по этому адресу заносится TRUE, в случае успеха

выборки, и FALSE в других случаях. В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства **LastErrorCode**, а тест ошибки - **LastErrorText**.

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

#include "PLDbConnector_i.c"
#include "PLDbConnector.h"    // для 32-х разрадной версии

...
CoInitialize(NULL);
...
// SMART указатель на объект с интерфейсом IDbRequest
CComPtr<IDbRequest> l_spIDbRequest;
HRESULT hr = l_spIDbRequest.CoCreateInstance(CLSID_DbRequest);

If (hr == S_OK) {
    // Задаем значения для некоторых свойств
    ...

    // Выполняем запрос
    BOOL fSuccess;
    hr = l_spIDbRequest->DoSelect(&fSuccess);

    if (hr != S_OK) {
        CComBSTR l_bstrErrorText;
        l_spIDbRequest->get_LastErrorText(&l_bstrErrorText);
        MessageBox(GetFocus(),  bstr t(l bstrErrorText),
                   "Ошибка запроса", MB_OK | MB_ICONSTOP);
    }
    else {
        // Запрашиваем результат выборки
        ...
    }
}
...
```

Асинхронная выборка

Альтернативой методу синхронной выборки является асинхронная выборка. Код программы, задав параметры выборки и выбрав один из механизмов «обратного вызова» запускает процесс выборки данных. Управление возвращается вызывающей программе. Для определения момента окончания выборки, программа может использовать один из трех вариантов «обратного вызова»:

- _CALLBACK_TYPE_NONE. Отсутствие механизма «обратного вызова». Этот вариант используется по умолчанию. Программа может определить окончание процесса выборки, только периодически проверяя значение свойства **AsyncSelectInProgress**. Когда значение свойства становится равным FALSE, то выборка завершена.
- _CALLBACK_TYPE_EVENT. Использование события (Event), которое сигнализирует об окончании асинхронной выборки.
- _CALLBACK_TYPE_WINDOW. Посылка окну приложения специального сообщения.

_CALLBACK_TYPE_NONE

Это самый простой механизм «обратного вызова». Точнее отсутствие механизма. Используется по умолчанию при создании объекта DbRequest. После запуска процесса асинхронной выборки

методом **AsyncSelect()** приложение может определить окончание процесса, только периодически проверяя значение свойства **AsyncSelectInProgress**.
Для задания механизма _CALLBACK_TYPE_NONE для объекта DbRequest используется метод SetCallbackNull().

SetCallbackNull

Задает для объекта механизм «обратного вызова» _CALLBACK_TYPE_NONE.

C++

```
HRESULT CDbRequest::SetCallbackNull();
```

Возвращает **S_OK** в случае успеха и **S_FALSE** в случае, когда процесс выборки в момент вызова метода еще не завершен. В этом случае свойство LastErrorCode возвратит значение _LE_SELECTION_IN_PROGRESS ("Асинхронная выборка еще выполняется").

```
...
// Задаем механизм обратного вызова _CALLBACK_TYPE_NONE
HRESULT hr = l_spIDbRequest.SetCallbackNull();

if (hr == S_OK) {
    // Запускаем выборку асинхронно
    VARIANT_BOOL bInProgress;
    hr = spIDbRequest->AsyncSelect(&bInProgress);
    if (hr == S_OK) {
        time_t tmStartTime, tmCurTime;
        time(&tmStartTime);

        // Ждем завершения выборки, но не более 10 секунд
        bInProgress = TRUE;
        BOOL bInterrupted = FALSE;
        do {
            spIDbRequest->get_AsyncSelectInProgress(&bInProgress);
            if (bInProgress) {
                time(&tmCurTime);
                if ((tmCurTime - tmStartTime > 10) && !bInterrupted) {
                    // Прерываем процесс выборки
                    spIDbRequest->StopAsyncSelect();
                    bInterrupted = TRUE;
                }
                Sleep(100);
            }
        } while(bInProgress);

        if (!bInterrupted) {
            LONG nLastErrorCode;
            spIDbRequest->get_LastErrorCode(&nLastErrorCode);

            if (nLastErrorCode == _LE_NO_ERROR) {
                // Получаем результат выборки
                ...
            }
        }
    }
}
else {
```

```
    // выборка выполняется в данный момент…
    ...
}
...
```

VBScript

```
'Задаем механизм обратного вызова _CALLBACK_TYPE_NONE
l_objDbRequest.SetCallbackNull
If l_objDbRequest.LastErrorCode = 0 Then
    'Запускаем выборку асинхронно
    If l_objDbRequest.AsyncSelect Then
        'Ждем завершения выборки
        While l_objDbRequest.AsyncSelectInProgress
        Wend

        If l_objDbRequest.LastErrorCode = 0 Then
            'Получаем результат выборки
            ...

        EndIf
    EndIf
EndIf
```

_CALLBACK_TYPE_EVENT

Механизм обратного вызова _CALLBACK_TYPE_EVENT имеет по сравнению с _CALLBACK_TYPE_NONE несколько преимуществ, несмотря на то, что он несколько сложнее:

- Позволяет ждать завершения процесса выборки в однопоточном приложении, без утилизации ресурсов процессора;
- Варьируя время ожидания, принимать решение о последующем ожидании или прерывании процесса выборки (метод StopAsyncSelect()).

Этот механизм обратного вызова основывается на использовании события (Event), которое получает состояние **Signaled** по окончании процесса выборка. Первое, что должно выполнить приложение – получить от объекта DbConnector дескриптор события. Дескриптор возвращается методом SetCallbackEvent().

SetCallbackEvent

Задает для объекта механизм обратного вызова _CALLBACK_TYPE_EVENT и возвращает дескриптор события.

C++

```
HRESULT CDbRequest::SetCallbackEvent(ULONG* pEvent);
```

Возвращает **S_OK** в случае успеха и **S_FALSE** в случае, когда процесс выборки в момент вызова метода еще не завершен. В этом случае свойство LastErrorCode возвратит значение _LE_SELECTION_IN_PROGRESS ("Асинхронная выборка еще выполняется").

pEvent

Выходной параметр. Адрес переменной, в которую будет занесено значение дескриптора события.

```
...
// Задаем механизм обратного вызова _CALLBACK_TYPE_EVENT
// и сохраняем дескриптор события
ULONG hEvent;
HRESULT hr = l_spIDbRequest.SetCallbackEvent(&hEvent);

if (hr == S_OK) {
    // Запускаем выборку асинхронно
}
else {
    // выборка выполняется в данный момент…
    ...
}
...
```

VBScript

```
'Задаем механизм обратного вызова _CALLBACK_TYPE_EVENT
' и сохраняем дескриптор события
Dim hEvent
hEvent = l_objDbRequest.SetCallbackEvent

If l_objDbRequest.LastErrorCode = 0 Then
    'Запускаем выборку асинхронно
Else
    ' выборка выполняется в данный момент…
EndIf
```

После получения приложением дескриптора события, оно должно запустить асинхронную выборку. Для определения момента окончания выборки приложение ждёт на событии изменения его состояния на Signaled.

WaitForCallbackEvent

```
HRESULT CDbRequest::WaitForCallbackEvent(
                [in] ULONG hEvent,
                [in] LONG lMilliseconds,
                [out, retval] ULONG* puRetValue);
```

Вспомогательный метод для скриптовых языков, не имеющих возможность вызова Windows API функций синхронизации WaitForSingleObject/WaitForMultipleObjects. Метод WaitForCallbackEvent представляет собой эквивалент WaitForSingleObject.

hEvent

Событие (Event объект), возвращаемое при вызове метода SetCallbackEvent;

lMilliseconds

Таймаут ожидания события, в миллисекундах. Значение таймаута может варьироваться от 0 (поверка состояния и возврат) до INFINITE (0xFFFFFFFF / -1) – бесконечное ожидание, до наступления события.

puRetValue

Возврат метода. Целое число без знака. Может принимать значения:

Возврат	Значение	Описание
WAIT_OBJECT_0	0	Event объект в состоянии signaled. Событие произошло.
WAIT_TIMEOUT	258	Истек таймаут ожидания. Событие не произошло.
WAIT_FAILED	0xFFFFFFFF	Event объект не существует.

Коду программы на C++ нет необходимости использовать метод WaitForCallbackEvent. Вместо этого он может использовать «штатные» функции синхронизации.

VBScript

```
'Объявляем переменную в которую метод SetCallbackEvent занесет Event
Dim l_hEvent
hEvent = l_objDbRequest.SetCallbackEvent

If l_objDbRequest.LastErrorCode = 0 Then
    'Запускаем выборку асинхронно
    If l_objDbRequest.AsyncSelect Then
        Dim nRetCode
        ' Ждем завершения процесса выборки. -1 означает бесконечное ожидание
        nRetCode = objDbRequest.WaitForCallbackEvent(hEvent, -1)
        If nRetCode = 0 And l_objDbRequest.LastErrorCode = 0 Then
            'Получаем результат выборки
            ...
        EndIf
    EndIf
Else
    ' выборка выполняется в данный момент…
EndIf
```

_CALLBACK_TYPE_WINDOW

Механизм обратного вызова через посылку сообщения окну программы. По окончании асинхронной выборки (но не самого процесса/потока выборки) в заданное окно посылается сообщение. Приложение, получив сообщение должно дождаться завершения процесса выборки, анализируя значение свойства AsyncSelectInProgress, до тех пор, пока оно не будет иметь значение FALSE. После этого приложение может перейти к получению результата выборки. Для использования механизма _CALLBACK_TYPE_WINDOW, до начала асинхронной выборки необходимо вызвать метод SetCallbackWindow().

SetCallbackWindow

```
HRESULT CDbRequest::SetCallbackWindow([in] HWND hwnd, [in] UINT uMsg);
```

hwnd

Handle окна приложения, которому будет направлено сообщение

uMsg

Номер сообщения. Рекомендуется использовать значения больше WM_USER. Значение параметра LPARAM будет содержать адрес объекта DbRequest, пославшего в окно сообщение.

Возвращает **S_OK** в случае успеха и **S_FALSE** в случае, когда процесс выборки в момент вызова метода еще не завершен. В этом случае свойство LastErrorCode возвратит значение _LE_SELECTION_IN_PROGRESS ("Асинхронная выборка еще выполняется").

Ниже приведен пример кода на C++ условного приложения, получающего сообщение через вызов функции окна.

C++

```
...
l_spIDbRequest->SetCallbackWindow(g_hwndMain, WM_USER+200);

BOOL bInProgress;
HRESULT hr = l_spIDbRequest->AsyncSelect(&bInProgress);

...
...

LRESULT CALLBACK MainWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch (msg) {
        ...

        case WM_USER+200:
        {
            IDbRequest *l_pIDbRequest = (IDbRequest *)lParam;
            BOOL bInProgress = TRUE;

            while(bInProgress) {
                l_spIDbRequest->get_AsyncSelectInProgress(&bInProgress);
                Sleep(10);
            }

            LONG l_lLastErrorCode;
             l_pIDbRequest->get_LastErrorCode(&l_lLastErrorCode);
             If (lLastErrorCode == _LE_NO_ERROR) {
                // Не было ошибки в процессе выборки
                // получаем результат выборки
                ...

             }
```

```
            return 0;
        }
        ...
        ...
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}
```

AsyncSelect

Запускает процесс асинхронной выборки данных, используя заданные параметры выборки (см. метод DoSelect) и текущий заданный механизм обратного вызова: _CALLBACK_TYPE_NONE, _CALLBACK_TYPE_EVENT или _CALLBACK_TYPE_WINDOW.

Метод возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. Если методу передается выходной параметр `pSuccess`, то в переменную по этому адресу заносится TRUE, в случае успеха запуска асинхронной выборки, и FALSE в других случаях. В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства **LastErrorCode**, а тест ошибки - **LastErrorText**. Возможны следующие ошибки:

- **_LE_SELECTION_IN_PROGRESS** (6) - "Асинхронная выборка еще выполняется"
- Другой код ошибки, отличный от 0 - "Ошибка создания потока: …"

C++

```
HRESULT CDbRequest::AsyncSelect([out, retval] BOOL* pSuccess);
```

```
...
// Задаем параметры выборки и механизм обратного вызова
...
BOOL bInProgress;
HRESULT hr = l_spIDbRequest->AsyncSelect(&bInProgress);

if (hr == S_OK) {
    // Асинхронная выборка в процессе работы
    // Ожидать завершения, в зависимости от механизма обратного вызова
   ...
}
else {
    // Ошибка запуска асинхронной выбрки. Получить код ошибки
    LONG l_lLastErrorCode;
     l_spIDbRequest->get_LastErrorCode(&l_lLastErrorCode);
     // Получить текст ошибки
     CComBSTR l_bstrErrorText;
     l_spIDbRequest->get_LastErrorText(&l_bstrErrorText);
    ...
}
...
```

VBScript

```
...
'Задаем параметры выборки и механизм обратного вызова
...
'Запускаем выборку асинхронно
If l_objDbRequest.AsyncSelect Then
    'Асинхронная выборка в процессе работы
    'Ожидать завершения, в зависимости от механизма обратного вызова
    ...
Else
    'Ошибка запуска асинхронной выбрки. Получить код ошибки
    Dim nErrorCode
    nErrorCode = l_objDbRequest.LastErrorCode
    'Получить текст ошибки
    Dim sErrorText
    sErrorText = l_objDbRequest.LastErrorText
    ...
EndIf
```

StopAsyncSelect

Дает сигнал на прерывание процесса асинхронной выборки данных. Процесс выборки не будет прерван моментально, поэтому приложение по-прежнему должно ожидать завершения процесса выборки.

C++

```
HRESULT CDbRequest::StopAsyncSelect();
```

```
...
// Задаем механизм обратного вызова _CALLBACK_TYPE_NONE
HRESULT hr = l_spIDbRequest.SetCallbackNull();

if (hr == S_OK) {
    // Запускаем выборку асинхронно
    VARIANT_BOOL bInProgress;
    hr = spIDbRequest->AsyncSelect(&bInProgress);
    if (hr == S_OK) {
        time_t tmStartTime, tmCurTime;
        time(&tmStartTime);

        // Ждем завершения выборки
        // Если выборка затянется на 10 секунд, то прерываем ее
        bInProgress = TRUE;
        BOOL bInterrupted = FALSE;
        do {
            spIDbRequest->get_AsyncSelectInProgress(&bInProgress);
            if (bInProgress) {
                time(&tmCurTime);
                if ((tmCurTime - tmStartTime > 10) && !bInterrupted) {
                    // Прерываем процесс выборки
                    spIDbRequest->StopAsyncSelect();
                    bInterrupted = TRUE;
                }
                Sleep(100);
```

```
            }
        } while(bInProgress);

        if (!bInterrupted) {
            LONG nLastErrorCode;
            spIDbRequest->get_LastErrorCode(&nLastErrorCode);

            if (nLastErrorCode == _LE_NO_ERROR) {
                // Получаем результат выборки
                ...
            }
        }
    }
}
...
```

VBScript

```
'Задаем механизм обратного вызова _CALLBACK_TYPE_NONE
l_objDbRequest.SetCallbackNull
If l_objDbRequest.LastErrorCode = 0 Then
    'Запускаем выборку асинхронно
    If l_objDbRequest.AsyncSelect Then
        Dim dtStart : dtStart = now
        Dim bInterrupted : bInterrupted = False

        'Ждем завершения выборки, но не более 10 секунд
        While l_objDbRequest.AsyncSelectInProgress
            If Not bInterrupted And (DateDiff("s", Now, dtStart) > 10) Then
                l_objDbRequest.StopAsyncSelect
                bInterrupted = True
            End If
        Wend

        If Not bInterrupted And l_objDbRequest.LastErrorCode = 0 Then
            'Получаем результат выборки
            ...
        EndIf
    EndIf
EndIf
```

Получение результатов выборки

В результате успешной выборки, вне зависимости синхронной или асинхронной, объект DbRequest сохраняет во внутренних структурах результы выборки (записи). Перед получением результатов выборки, необходимо убедиться что LastErrorCode объекта равен нулю. Далее необходимо запросить число выбранных записей используя свойство ResultRecordCount. В цикле запрашивать записи с индексом от 0 до числа, возвращенного ResultRecordCount – 1. Запись передается приложению в виде объекта ResultRecord.

GetResultRecord

```
HRESULT CDbRequest::GetResultRecord(
    [in] LONG lIndex,
    [out, retval] IResultRecord** ppRecord);
```

Создает объект **ResultRecord** по заданному индексу массива выбранных по запросу записей.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки устанавливается в ERROR_INVALID_PARAMETER, а текст ошибки в "Неверно задан индекс массива".

lIndex

Индекс массива выбранных записей. Значение запрашиваемого индекса должно находиться в диапазоне значений от 0 до значения, свойства **`ResultRecodCount`** минус 1.

ppRecord

Адрес указателя на интерфейс создаваемого объекта. При выходе указатель будет заполнен значением.

Объекты **ResultRecord** в коде внешнего ПО **не создаются напрямую**. Вместо этого, у объекта DbRequest запрашивается элемент массива выбранных записей. Полученный от метода объект (интерфейс объекта) **должен быть удален в коде приложения**.

VBScript

```
'Выполняем запрос
l_objDbRequest.DoSelect

If l_objDbRequest.LastErrorCode = 0 Then
    'Запрашиваем число выбранных записей
    Dim l_nRecordCount : l_nRecordCount = l_objDbRequest.ResultRecordCount
    If l_nRecordCount > 0 Then
        Dim l_objCurrentRecord
        'Запрашиваем объект с индексом 0
        Set l_objCurrentRecord = l_objDbRequest.GetResultRecord(0)
        'Запрашиваем свойства объекта ResultRecord
        Dim l_dt : l_dt = l_objCurrentRecord.DateTime
        ...
        'Удаляем объект
        Set l_objCurrentRecord = Nothing
    End If
End If
...
```

C++

```
...
// Выполняем запрос
hr = l_spIDbRequest->DoSelect();

if (hr == S_OK) {
    // Запрашиваем число элементов массива выбранных записей
    LONG l_lRecordCount = 0;
    l_spIDbRequest->get_ResultRecordCount(&l_lRecordCount);

    if (l_lRecordCount > 0) {
        CComPtr <IResultRecord> l_spCurrentRecord;
```

```
            for (int i=0; i < l_lRecordCount; i++) {

                // Запрашиваем интерфейс объекта с индексом i
                if (S_OK == l_spIDbRequest->GetResultRecord(
                                        i, &l_spCurrentRecord))
                {
                    // Запрашиваем свойства объекта CurrentRecord
                    DATE l_dt;
                    l_spCurrentRecord->get_DateTime(&l_dt);
                    ...
                    // Удаляем объект, чтобы использовать в цикле повторно
                   l_spCurrentRecord = NULL;
                }
            }   // for
        }
    }
    ...
```

Объект ResultRecord

После выполнения выборки данных в объекте DbRequest содержится коллекция (массив записей) объектов класса **ResultRecord** с результатами выборки. Объекты **ResultRecord** не создаются напрямую. Они запрашиваются у объекта DbRequest с помощью метода **GetResultRecord**.
Количество элементов в коллекции содержится в свойстве **ResultRecordCount** .
Объект имеет набор свойства, и один метод **GetAttibute**, возвращающий атрибут с заданным индексом из коллекции атрибутов объекта. Объект имеет единственный интерфейс **IResultRecord**.
Все свойства объекта имеют атрибуты READ ONLY, т.е. могут быть запрошены, но не могут быть изменены.

Свойства объекта ResultRecord

DateTime

Вещественное число двойной точности. Содержит дату и время события ответа клиента на вопрос.

VBScript

```
If l_objDbRequest.DoSelect Then
    'Запрашиваем число выбранных записей
    Dim l_nRecordCount : l_nRecordCount = l_objDbRequest.ResultRecordCount
    Dim i

    For i=0 To l_nRecordCount
        Dim l_objCurrentRecord
        'Запрашиваем объект с индексом i
        Set l_objCurrentRecord = l_objDbRequest.GetResultRecord i
        'Запрашиваем свойства объекта ResultRecord
        Dim l_dt : l_dt = l_objCurrentRecord.DateTime
        ...
        'Удаляем объект. Не обязательно
        Set l_objCurrentRecord = Nothing
    Next
End If
...
```

C++

```
HRESULT CResultRecord::get_DateTime([out, retval] DATE* pValue);
```

```
...
// Выполняем запрос
hr = l_spIDbRequest->DoSelect();

if (hr == S_OK) {
    // Запрашиваем число элементов массива выбранных записей
    LONG l_lRecordCount = 0;
    l_spIDbRequest->get_ResultRecordCount(&l_lRecordCount);

    for (LONG i=0; i < l_lRecordCount; i++)
```

```
    {
        CComPtr <IResultRecord> l_spCurrentRecord;

        // Запрашиваем интерфейс объекта с индексом i
        if (S_OK == l_spIDbRequest->GetResultRecord(i, &l_spCurrentRecord))
        {
            // Запрашиваем дату/время события
            DATE l_dt;
            l_spCurrentRecord->get_DateTime(&l_dt);
            ...
            // Удаляем объект, чтобы использовать в цикле повторно
            l_spCurrentRecord = NULL;
        }
    }
}
......
```

Location

Возвращает Концентратор Данных – компьютер от которого информация об ответе клиента на вопрос была передана в базу данных. Строка.

VBScript

```
Dim l_objCurrentRecord
'Запрашиваем объект с индексом i
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord i
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sLocation: l_sLocation = l_objCurrentRecord.Location
...
```

C++

```
HRESULT CResultRecord::get_Location([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord;

// Запрашиваем интерфейс объекта с индексом i
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);

    // Запрашиваем Location
    CComBSTR l_bstrLocation;
    l_spCurrentRecord->get_Location(&l_bstrLocation);
    ...
    l_spCurrentRecord = NULL;
}
...
```

Panel

Возвращает имя пульта на котором была нажата кнопка ответа. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sLocation: l_sLocation = l_objCurrentRecord.Location
Dim l_sPanel: l_sPanel = l_objCurrentRecord.Panel
...
```

C++

```
HRESULT CResultRecord::get_Panel([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта с индексом i
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);

    // Запрашиваем Location
    CComBSTR l_bstrLocation;
    l_spCurrentRecord->get_Location(&l_bstrLocation);

    // Запрашиваем Panel
    CComBSTR l_bstrPanel;
    l_spCurrentRecord->get_Panel(&l_bstrPanel);
    ...
    l_spCurrentRecord = NULL;
}
...
```

Question

Возвращает текст вопроса, на который отвечал клиент. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sLocation: l_sLocation = l_objCurrentRecord.Location
Dim l_sPanel: l_sPanel = l_objCurrentRecord.Panel
Dim l_sQuestion: l_sQuestion = l_objCurrentRecord.Question
...
```

C++

```
HRESULT CResultRecord::get_Question([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта с индексом i
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);

    // Запрашиваем Location
    CComBSTR l_bstrLocation;
    l_spCurrentRecord->get_Location(&l_bstrLocation);

    // Запрашиваем Panel
    CComBSTR l_bstrPanel;
    l_spCurrentRecord->get_Panel(&l_bstrPanel);

    // Запрашиваем текст вопроса
    CComBSTR l_bstrQuestion;
    l_spCurrentRecord->get_Question(&l_bstrQuestion);
    ...
    l_spCurrentRecord = NULL;
}
...
```

AnswerAlias

Возвращает псевдоним варианта ответа, который дал клиент на заданный вопрос. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sLocation: l_sLocation = l_objCurrentRecord.Location
Dim l_sPanel: l_sPanel = l_objCurrentRecord.Panel
Dim l_sQuestion: l_sQuestion = l_objCurrentRecord.Question
Dim l_sAnswerAlias: l_sAnswerAlias = l_objCurrentRecord.AnswerAlias
...
```

C++

```
HRESULT CResultRecord::get_AnswerAlias([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord;
```

```
// Запрашиваем интерфейс объекта с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);

    // Запрашиваем Location
    CComBSTR l_bstrLocation;
    l_spCurrentRecord->get_Location(&l_bstrLocation);

    // Запрашиваем Panel
    CComBSTR l_bstrPanel;
    l_spCurrentRecord->get_Panel(&l_bstrPanel);

    // Запрашиваем текст вопроса
    CComBSTR l_bstrQuestion;
    l_spCurrentRecord->get_Question(&l_bstrQuestion);

    // Запрашиваем псевдоним варианта ответа
    CComBSTR l_bstrAnswerAlias;
    l_spCurrentRecord->get_AnswerAlias(&l_bstrAnswerAlias);
    ...
    l_spCurrentRecord = NULL;
}
...
```

UserName

Возвращает имя пользователя компьютера, на котором клиент дал ответ на заданный вопрос.

Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
...
Dim l_sUserName: l_sUserName = l_objCurrentRecord.UserName
...
```

C++

```
HRESULT CResultRecord::get_UserName([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord;

// Запрашиваем интерфейс объекта с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
```

```
    l_spCurrentRecord->get_DateTime(&l_dt);
    ...

    // Запрашиваем UserName
    CComBSTR l_bstrUserName;
    l_spCurrentRecord->get_UserName(&l_bstrUserName);
    ...
    l_spCurrentRecord = NULL;
}
...
```

UserDomain

Возвращает домен/рабочую группу пользователя компьютера, на котором клиент дал ответ на заданный вопрос. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
...
Dim l_sUserDomain: l_sUserDomain = l_objCurrentRecord.UserDomain
...
```

C++

```
HRESULT CResultRecord::get_UserDomain([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord;

// Запрашиваем интерфейс объекта с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);
    ...

    // Запрашиваем UserDomain
    CComBSTR l_bstrUserDomain;
    l_spCurrentRecord->get_UserDomain(&l_bstrUserDomain);
    ...
    l_spCurrentRecord = NULL;
}
...
```

UserDepartment

Возвращает название подразделения пользователя компьютера, на котором клиент дал ответ на заданный вопрос. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
...
Dim l_sUserDomain: l_sUserDomain = l_objCurrentRecord.UserDomain
...
```

C++

```
HRESULT CResultRecord::get_UserDomain([out, retval] BSTR* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord;

// Запрашиваем интерфейс объекта с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);
    ...

    // Запрашиваем UserDomain
    CComBSTR l_bstrUserDomain;
    l_spCurrentRecord->get_UserDomain(&l_bstrUserDomain);
    ...
    l_spCurrentRecord = NULL;
}
...
```

AttributeCount

Число. Возвращает количество элементов коллекции аттрибутов в записи (объекте ResultRecord). Атрибут это пара строковых значений <**Имя атрибута**> и <**Значение атрибута**>. Например <ID клента> = <Иванов П.М.>.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом i
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord i
'Запрашиваем число элементов в коллекции атрибутов
```

```
Dim l_nAttrCount
l_nAttrCount = l_objCurrentRecord.AttributeCont
If nAttrCount > 0 Then
    'Запрашиваем имя и значение атрибута с индексом 0
    Dim l_sAttrName
    l_sAttrName = l_objCurrentRecord. GetAttributeName(0)
    Dim l_sAttrValue
    l_sAttrValue = l_objCurrentRecord. GetAttributeValue(0)
    ...
EndIf
...
```

C++

```
HRESULT CResultRecort::get_AttributeCount([out, retval] LONG* pValue);
```

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта выборки с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    ...
    // Запрашиваем число элементов в коллекции атрибутов
    UINT l_uAttrCount;
    l_spCurrentRecord->get_AttributeCount(&l_uAttrCount);

    if (l_uAttrCount) {
        // Запрашиваем имя и значение атрибута с индексом 0
        CComBSTR l_bstrAttrName, l_bstrAttrValue;
        l_spCurrentRecord->get_GetAttr(0, &l_bstrAttrName, &l_bstrAttrValue);
        ...
    }
    ...
}
...
```

Методы объекта ResultRecord

GetAttributeName

```
HRESULT CResultRecord::GetAttributeName(
    [in] LONG lIndex,
    [out] BSTR *pAttrName
);
```

Возвращает имя атрибута объекта **ResultRecord** по заданному индексу атрибута в коллекции.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки устанавливается в ERROR_INVALID_PARAMETER, а текст ошибки в "Неверно задан индекс массива".

lIndex

Индекс атрибута в коллекции (массиве) атрибутов. Значение запрашиваемого индекса должно находиться в диапазоне значений от 0 до значения, свойства **`AttrColSize`** минус 1.

pAttrName

Адрес BSTR объекта, в который будет помещено имя атрибута.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем число атрибутов в коллекции
Dim nAttrCount : nAttrCount = l_objCurrentRecord.GetAttributeCount
Dim i
For i=0 To nAttrCount -1
   Dim l_AttrName
   l_AttrName = l_objCurrentRecord.GetAttributeName(i)
   ...
Next
...
```

C++

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта выборки с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    ...
    // Запрашиваем число элементов в коллекции атрибутов
    UINT l_uAttrCount;
    l_spCurrentRecord->get_AttributeCount(&l_uAttrCount);

    // В цикле запрашиваем имена атрибутов
    for ( UINT i=0; i < l_uAttrCount; i++)
    {
        // Запрашиваем имя атрибута с индексом i
        CComBSTR l_bstrAttrName;
        l_spCurrentRecord->get_GetAttributeName(0, &l_bstrAttrName);
        ...
    }
    ...
}
...
```

GetAttributeValue

```
HRESULT CResultRecord::GetAttributeValue(
    [in] LONG lIndex,
    [out] BSTR *pAttrVaue
);
```

Возвращает значение атрибута объекта **ResultRecord** по заданному индексу атрибута в коллекции.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки устанавливается в ERROR_INVALID_PARAMETER, а текст ошибки в "Неверно задан индекс массива".

lIndex

Индекс атрибута в коллекции (массиве) атрибутов. Значение запрашиваемого индекса должно находиться в диапазоне значений от 0 до значения, свойства **`AttrColSize`** минус 1.

pAttrValuue

Адрес BSTR объекта, в который будет помещено значение атрибута.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDbRequest.GetResultRecord 0
'Запрашиваем число атрибутов в коллекции
Dim nAttrCount : nAttrCount = l_objCurrentRecord.GetAttributeCount
Dim i
For i=0 To nAttrCount -1
   Dim l_AttrValue
   l_AttrValue = l_objCurrentRecord.GetAttributeValue(i)
   ...
Next
...
```

C++

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта выборки с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    ...
    // Запрашиваем число элементов в коллекции атрибутов
    UINT l_uAttrCount;
    l_spCurrentRecord->get_AttributeCount(&l_uAttrCount);

    // В цикле запрашиваем значения атрибутов
    for ( UINT i=0; i < l_uAttrCount; i++)
    {
        // Запрашиваем значение атрибута с индексом i
        CComBSTR l_bstrAttrValue;
        l_spCurrentRecord->get_GetAttributeValue(0, &l_bstrAttrValue);
        ...
    }
    ...
}
...
```

GetAttr

```
HRESULT CResultRecord::GetAttr(
    [in] LONG lIndex,
    [out] BSTR *pAttrName,
    [out] BSTR *pAttrValue);
```

Возвращает имя и значение атрибута объекта **ResultRecord** по заданному индексу атрибута в коллекции.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки устанавливается в ERROR_INVALID_PARAMETER, а текст ошибки в "Неверно задан индекс массива".

lIndex

Индекс атрибута в коллекции (массиве) атрибутов. Значение запрашиваемого индекса должно находиться в диапазоне значений от 0 до значения, свойства **AttributeCount** минус 1.

pAttrName

Адрес BSTR объекта, в который будет помещено имя атрибута.

pAttrValue

Адрес BSTR объекта, в который будет помещено значение атрибута.

VBScript

Метод не может использоваться в скриптовых языках.

C++

```
...
CComPtr <IResultRecord> l_spCurrentRecord = NULL;

// Запрашиваем интерфейс объекта выборки с индексом 0
if (S_OK == l_spIDbRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    ...
    // Запрашиваем число элементов в коллекции атрибутов
    UINT l_uAttrCount;
    l_spCurrentRecord->get_AttributeCount(&l_uAttrCount);

    // В цикле запрашиваем имена и значения атрибутов
    for ( UINT i=0; i < l_uAttrCount; i++)
    {
        // Запрашиваем имя и значение атрибута с индексом 0
        CComBSTR l_bstrAttrName, l_bstrAttrValue;
        l_spCurrentRecord->get_GetAttr(0, &l_bstrAttrName, &l_bstrAttrValue);
        ...
```

```
    }
    . . .
}
. . .
```

Время жизни модуля и объектов

Модуль COM-компонента (PLDbConnector.dll либо PLDbConnector64.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.

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: PLDbConnector.Request для 32-х разрядного кода и PLDdConnector64.Request 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.