2023

ProLAN

ProLAN LogConnector 1.0 32 & 64 bit

Руководство разработчика

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

Оглавление

Введение

Компонент ProLAN LogConnector (далее **PLLogConnector**) предназначен для выполнения запросов к LOG файлу, содержащему информацию о результатах ответов клиентов на различные вопросы. LOG файл формируется программой EPM-Agent Plus в заданной папке Windows компьютера или сервера локальной сети. LOG файл имеет формат CSV файла (значения, разделяемые символом табуляции) и, по сути, представляет собой таблицу со строками (записями) событий. Набор полей (столбцов таблицы) определяет некоторые параметры события.

Несмотря на некоторые ограничения возможности запроса данных, хранящихся в LOG файле, в частности доступ к данным возможен только для компьютеров в пределах локальной/корпоративной/виртуальной частной сети, LOG файл представляет экономичную альтернативу хранения результатов в полноценной базе данных или облачном сервисе. К недостаткам LOG файла также можно отнести ограничение на объем хранящихся в нем данных. При достижении размера 20 МБ, файл ротируется, и создается новый экземпляр текущего файла. Таким образом, объем данных колеблется в пределах 20-40 МБ. Тем не менее, такого объема достаточно для хранения сотен тысяч записей.

Развертывание решения с использованием агрегации данных на основе LOG файла на порядок дешевле и проще, что является безусловным достоинством. Данные LOG файла могут визуализироваться в режиме реального времени. По результатам, хранящимся в LOG файле, могут строиться отчеты.

Для интеграции с CRM (в частности 1С:Предприятие и т.п.) компания ProLAN разработала COM компонент PLLogConnector, который позволяет быстро и удобно выполнять запросы по интересующим параметрам и получать результаты выборки данных из LOG файла. Данные выборки могут быть использованы для импорта во внутренние хранилища/базы данных или создания отчетов «на лету».

Компонент PLLogConnector представляет собой 32-х или 64-х разрядный COM (ActiveX) компонент, который устанавливается на любой компьютер локальной сети, с которого возможен доступ к папке размещения LOG-файла. 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 11. В зависимости от разрядности кода приложений, использующих компонент необходимо устанавливать 32-х либо 64-х разрядную версию компонента. В 64-х разрядной ОС Windows могут быть установлены обе версии компонента.

Дистрибутивы установки компонента PLLogConnector вы можете загрузить по ссылкам:
https://www.prolan.ru/files/freetools/LogConnectorSetup.exe
https://www.prolan.ru/files/freetools/LogConnector64Setup.exe

Скачайте дистрибутив и запустите на выполнение файл LogConnectorSetup.exe или LogConnector64Setup.exe (потребуются права администратора). Последовательно проходите по страницам Мастера установки.

На странице Сведений о пользователе (рис.1), 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.

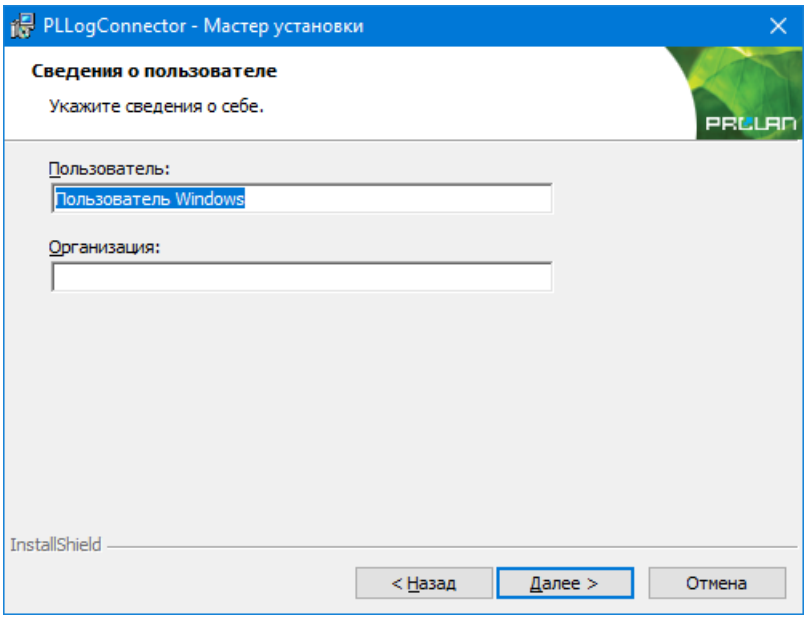

Рис.1. Сведения о пользователе

На следующей странице установки Вид установки (рис.2), 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 папки размещения LOG файла 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.

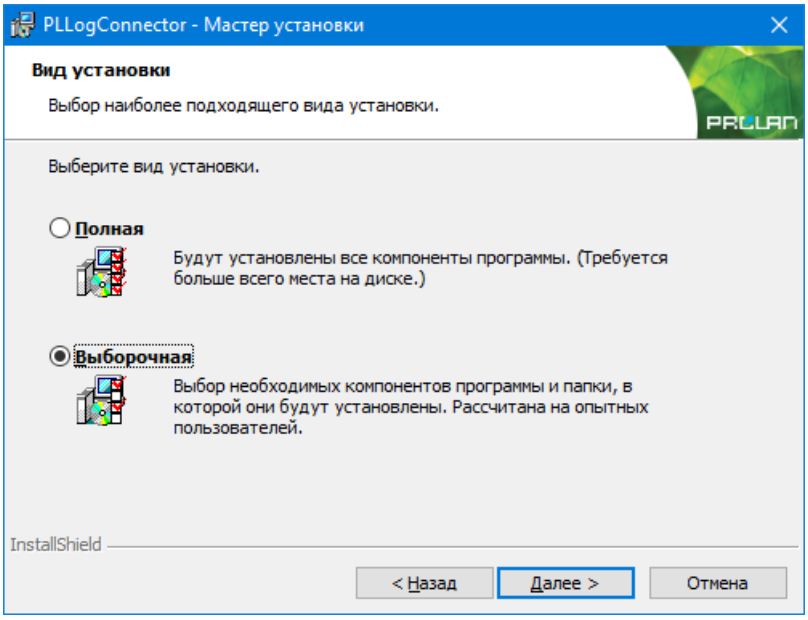

Рис.2. Вид установки

Если вы устанавливаете продукт на компьютере конечного пользователя, или хотите изменить папку установки по умолчанию, то выберите **Выборочная**, и на странице Выборочная установка (рис.3) произведите необходимые действия.

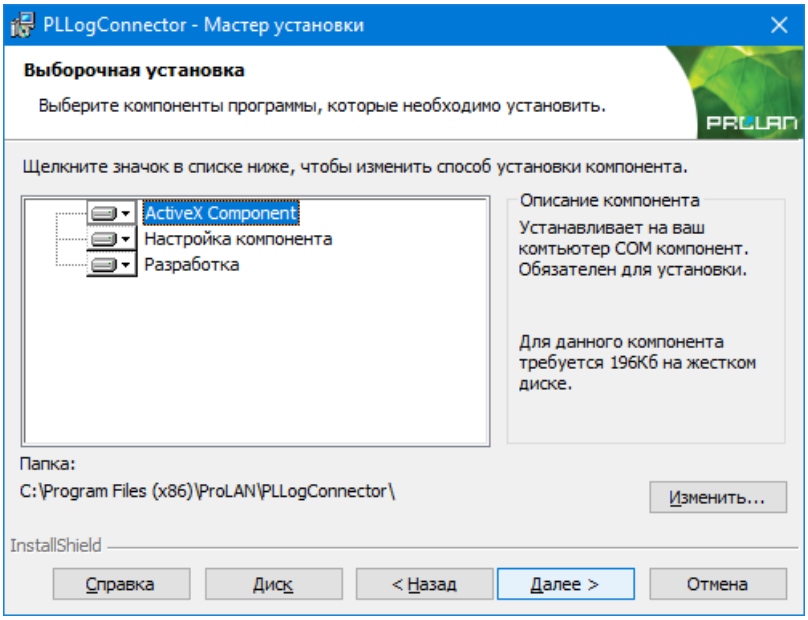

Рис.3. Выборочная установка

На рисунке 3 показана папка по умолчанию C:\Program Files (x86)\ProLAN\PLLogConnector, 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\ProLAN\PLLogConnector
Если это необходимо, то нажав кнопку **Изменить…**, вы можете задать другую папку установки.

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

Нажмите на кнопки **Далее** и **Установить** для начала процесса установки. По окончании установки в меню Windows **Пуск → Все программы** будет добавлена папка **ProLAN → LogConnector** или **LogConnector64** со значками запуска программ настройки COM компонента и просмотра данного руководство разработчика.

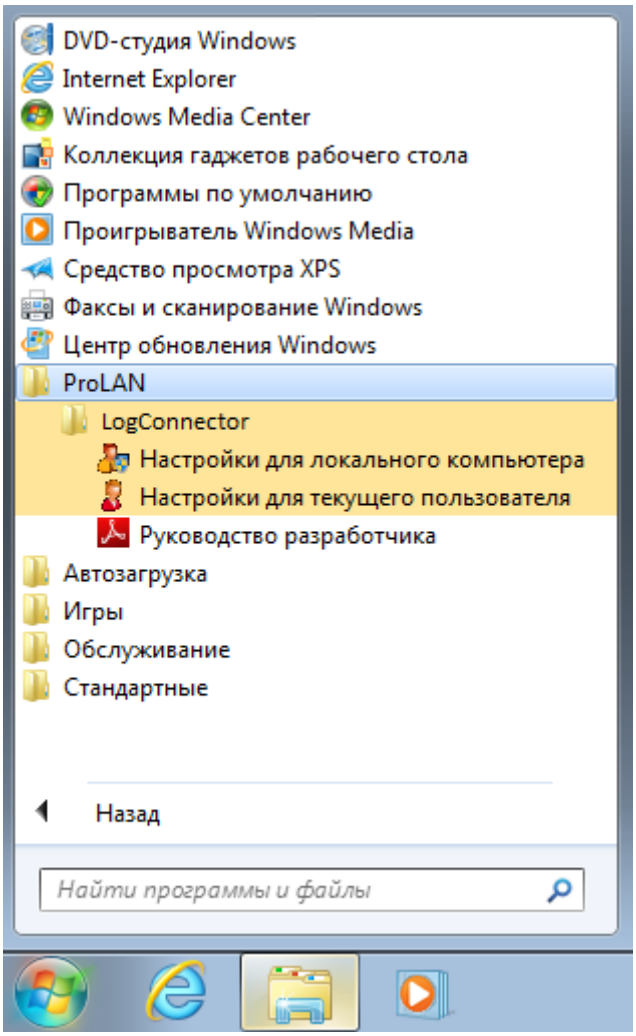

Рис.4. Пункты меню Window для запуска программ настройки компонента.

Настройки компонента

Для выполнения запросов к LOG файлу, COM компоненту необходимо сообщить **папку размещения LOG файла**. Папка может быть сообщена компоненту непосредственно в коде приложения, использующего компонент. Но прямое задание того параметра в коде приложения приводит к дополнительным затратам разработчика, хотя непосредственно к выполнению запросов отношения не имеют. Поэтому, в большинстве случаев удобно выполнить

предварительную настройку COM компонента и папку размещения LOG файла 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 (папка размещения LOG файла) могут храниться в реестре системы на двух уровнях:

- **Уровень текущего пользователя**. Значение параметров настроек 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 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 PLLogConnector, 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 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 5 показано окно диалога программы сразу после запуска.

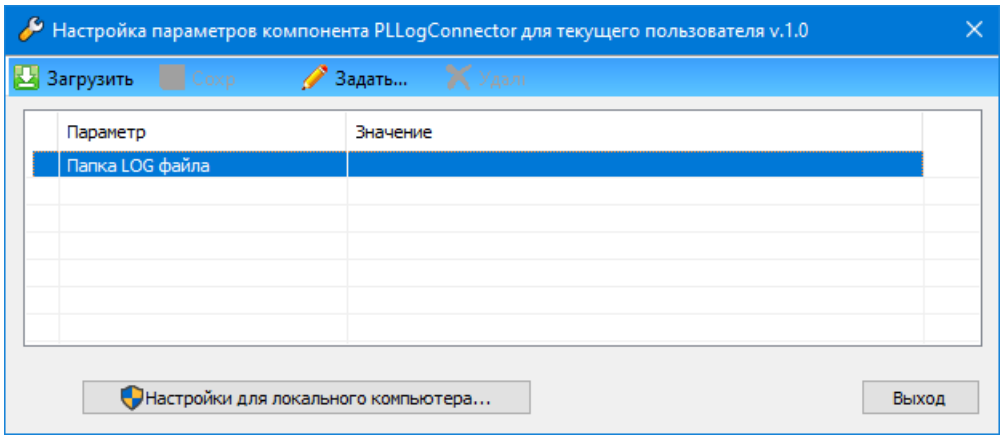

Рис.5. Окно диалога программы настроек для текущего пользователя.

Выберите в списке параметров **“Папка LOG файла”** и нажмите панели инструментов кнопку **“Задать”**.

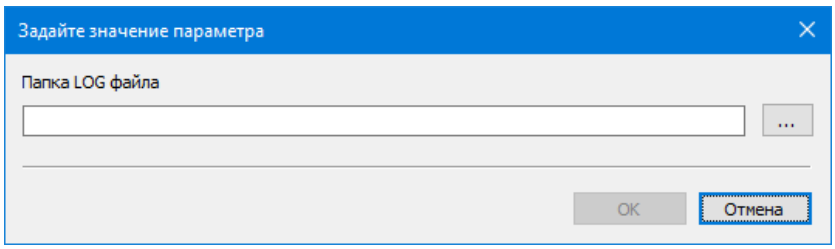

Рис.6. Диалог задания значения параметра.

В окне диалога (рис.6), нажмите на кнопку с тремя точками, и выберите папку LOG файла в стандартном диалоге Windows «Обзор компьютеров».

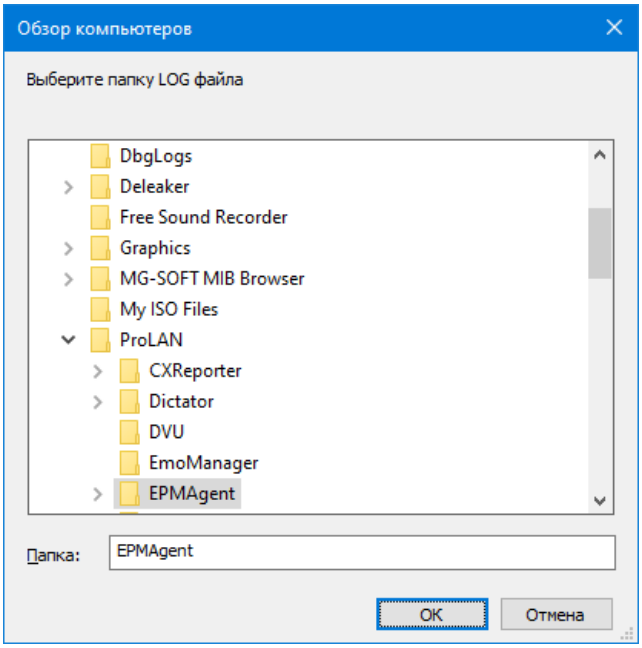

Рис.7. Выбор папки размещения LOG файла.

LOG файл формируется программой EPM-Agent Plus в папке, заданной в его настройках. По умолчанию, такой папкой является папка **<Документы>\ProLAN\EPMAgent**, связанная с текущим пользователем компьютера. Однако папка может быть изменена, и настроена, например, на папку файлового сервера. Выберите папку LOG файла (локальную папку компьютера или разделяемую в сети папку) и нажмите **OK**.

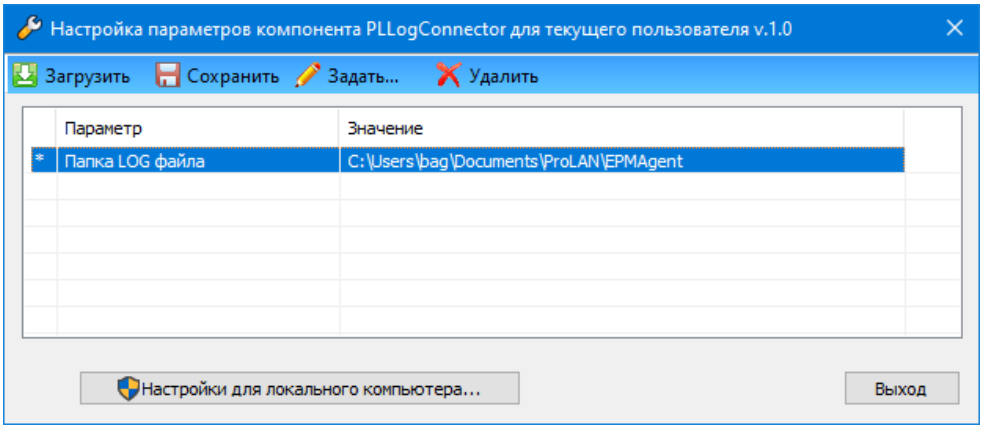

Рис.8. Папка LOG файла задана.

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 DataRequest

Создание объекта DataRequest в коде приложения

В качестве языка примеров, в данном руководстве будут 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.
- Встроенный язык программирования 1С:Предприятие 64-х разрядной версии.
- Примеры на C++.

Основным объектом компонента PLLogConnector является объект класса **CDataRequest** с интерфейсом **IDataRequest**. С помощью него выполняются запросы к LOG файлу и выборка результата.

VBScript

В скриптовых языках объект создается с использованием строкового эквивалента идентификатора интерфейса (ProgID), имеющего значение `PLLogConnector.DataRequest.1` либо `PLLogConnector64.DataRequest` для 32-х и 64-х разрядной версии компонента:

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

'Переменная будет содержать объект
Dim l_objDataRequest
'Создаем объект для 32-х разрядной версии компонента.
Set l_objDataRequest = CreateObject("PLLogConnector.DataRequest.1")
'Для 64-х разрядной версии компонента создание объекта выглядит так
' Set l_objDataRequest = CreateObject("PLLogConnector64.DataRequest.1")

'Объект создан?
If Err.number <> 0 Then
    MsgBox "Ошибка создания объекта 'PLLogConnector.DataRequest.1': " & _
           Err.Description, vbOKOnly + vbCritical, "Ошибка"
    Exit Sub
Else
    MsgBox "Объект создан.", vbOKOnly + vbInformation, _
           "Версия компонента " & l_objDataRequest.SVersion
    'Удаляем объект
    Set l_objDataRequest = Nothing
End If
```

1С

```
Попытка

  // Создать объект
  СоединениеLogФайлом = Новый COMобъект("PLLogConnector64.DataRequest.1");

Исключение
  Сообщить("Не удалось создать объект PLLogConnector64.DataRequest: " +
                  ОписаниеОшибки());
  Возврат;

КонецПопытки;

Сообщить("Объект PLLogConnector64.DataRequest создан. Версия: " +
          СоединениеLogФайлом.SVersion);


СоединениеLogФайлом = Неопределено;
```

C++

Подключите в код программы файлы "PLLogConnector_i.c" и " PLLogConnector_i.h" (PLLogConnector64_i.c и PLLogConnector64_i.h для 64-х разрядной версии), которые вы можете найти в подкаталоге CPP папки установки. Подключите заголовочный файл PLLogConnectorExt.h с кодами ошибок. Инициализируйте библиотеку OLE в коде потока.

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

#include "PLLogConnector_i.c"
#include "PLLogConnector_i.h" // для 32-х разрадной версии
// либо
#include "PLLogConnector64_i.c"
#include "PLLogConnector64_i.h"     // для 64-х разрадной версии

#include "PLogConnectorExt.h"

...

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

Объект DataRequest является потокобезопасным. Это означает, что создав объект в одном из потоков программы, вы можете использовать его 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 DataRequest:

- Задайте нужный вам для выборки набор свойств объекта:
  - Папка размещения LOG файла (опционально)
  - Начальная и конечная даты выборки
  - Другие свойства (опционально);
- 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 DataRequest

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_objDataRequest.SVersion
```

1C

```
СоединениеЛогФайлом = Новый COMобъект("PLLogConnector64.DataRequest.1");

Сообщить("Объект PLLogConnector64.DataRequest создан. Версия: " +
          СоединениеЛогФайлом.SVersion);
```

C++

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrVersion;
hr = l_spIDataRequest->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_SELECTION_IN_PROGRESS	4	Запрос выполняется в данный момент
_LE_LOG_FILE_FOLDER_NOT_SET	5	Не задана папка LOG файла
_LE_LOG_FILE_FOLDER_NOT_EXIST	6	Папка LOG файла не существует или недоступна
_LE_SELECTION_ERROR	7	Ошибка в процессе запроса или выборки данных
_LE_SELECTION_INTERRUPTED_BY_USER	8	Выполнение запроса было прервано пользователем
_LE_OPEN_LOG_FILE_ERROR	9	Ошибка открытия LOG-файла в заданной папке
_LE_READ_LOG_FILE_ERROR	10	Ошибка чтения LOG-файла
_LE_LOG_FILE_WRONG_FORMAT	11	LOG-файл имеет неверный формат
_LE_ASYNC_SELECTION_IN_PROGRESS	12	Асинхронный запрос на выборку выполняется в данный момент
ERROR_INVALID_PARAMETER	87	Неверно задан индекс массива

VBScript

```
Const _LE_NO_ERROR = 0
Const _LE_OUT_OF_MEMORY = 1
Const _LE_INVALIDARG = 2
Const _LE_UNKNOWN_ERROR = 3
Const _LE_SELECTION_IN_PROGRESS = 4
Const _LE_LOG_FILE_FOLDER_NOT_SET = 5
Const _LE_LOG_FILE_FOLDER_NOT_EXIST = 6
Const _LE_SELECTION_ERROR = 7
Const _LE_SELECTION_INTERRUPTED_BY_USER = 8
Const _LE_OPEN_LOG_FILE_ERROR = 9
Const _LE_READ_LOG_FILE_ERROR = 10
Const _LE_LOG_FILE_WRONG_FORMAT = 11
Const _LE_ASYNC_SELECTION_IN_PROGRESS = 12

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

1C

```
КодПоследнейОшибки = СоединениеЛогФайлом.LastErrorCode
```

C++

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

```
#include "PLLogConnectorExt.h"
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
LONG l_lLastErrorCode;
hr = l_spIDataRequest->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_objDataRequest.LastErrorText
```

1C

```
ТекстПоследнейОшибки = СоединениеЛогФайлом.LastErrorText
```

C++

```
HRESULT CDataRequest::get LastErrorText(BSTR* pErrorText);
```

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrErrorText;
hr = l_spDataRequest->get_LastErrorText(&l_bstrErrorText);
// В l_bstrErrorText теперь находится описание ошибки последней опрации
...
```

LogFileFolder

Получает или задает папку размещения LOG файла. Строка. Если были выполнены предварительные настройки компонента, то при создании объекта свойство LogFileFolder автоматически получит значение, сохраненное в реестре компьютера на уровне локального пользователя (более высокий приоритет) или локального компьютера. В противном случае LogFileFolder будет пустой строкой. Папка размещения LOG файла должна быть задана до выполнения запроса методом **DoSelect()** или **AsyncSelect()**.

При задании значения свойства, внутреннее значение в объекте может не измениться, если задаваемая строка будет иметь длину более 255 символов. В этом случае, код последней ошибки будет установлен в _LE_INVALIDARG, а текст последней ошибки - "Слишком большая длина строки".

VBScript

```
Dim l_sOldLogFileFolder
' Получаем текущее значение
l_sOldLogFileFolder = l_objDataRequest.LogFileFolder

If l_sOldLogFileFolder = "" then
  ' Задаем папку на файловом сервере
  l_objDataRequest.LogFileFolder = "z:\SharedFolder\EPMAgent"
EndIf
```

1C

```
ПапкаЛогФайла = СоединениеЛогФайлом.LogFileFolder

Если СтрДлина(ПапкаЛогФайла) = 0 Тогда
    СоединениеЛогФайлом.LogFileFolder = "\\FileServer\SharedFolder\EPMAgent";
КонецЕсли;
```

C++

```
HRESULT CDataRequest::get_LogFileFolder(BSTR* pCurrentValue);

HRESULT CDataRequest::put_LogFileFolder(BSTR newValue);
```

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrLogFileFolder;
hr = l_spIDataRequest->get_LogFileFolder(&bstrLogFileFolder);
// В l_bstrLogFileFolder теперь находится строка соединения
// Задаем папку на сервере
hr = l_spIDataRequest->put_LogFileFolder(
            CComBSTR(_T("z:\\SharedFolder\\EPMAgent")));
...
```

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_objDataRequest.BeginDate
' Задаем другую дату – час назад от текущего времени
```

```
l_objDataRequest.BeginDate = DateAdd("h", -1, now)
```

1C

Использовать свойство в 1С проблематично, т.к. тип значения «Дата» в 1С не соответствует типу DATE, которое используется в этом свойстве.
Используйте свойство BeginDateText, которое приемлемо для 1С.

C++

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

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

```
...
CComPtr<IDataRequest > l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
DATE l_dblOldBeginDate;
hr = l_spIDataRequest->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_spIDataRequest->put_BeginDate(dblDate);
if (hr != S_OK) {
      LONG l_lLastErrorCode;
      hr = l_spIDataRequest->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_objDataRequest.EndDate
' Задаем другую дату – час назад от текущего времени
l_objDataRequest.BeginDate = DateAdd("h", -1, now)
```

1C

Использвать свойство в 1С проблематично, т.к. тип значения «Дата» в 1С не соответствует типу DATE, которое используется в этом свойстве. Используйте свойство EndDateText, которое приемлемо для 1С.

C++

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

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

```
...
CComPtr<IDataRequest > l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
DATE l_dblOldEndDate;
hr = l_spIDataRequest->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_spIDataRequest->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_OUT_OF_MEMORY — Недостаточно памяти
- _LE_UNKNOWN_ERROR — Ошибка при преобразовании в тип DATA

VBScript

```
Dim l_sBeginDateText
' Получаем дату/время начала диапазона выборки данных
l_sBeginDateText = l_objDataRequest.BeginDateText
' Задаем другую дату
l_objDataRequest.BeginDateText = "2023-07-10 15:00:00"
If l_objDataRequest.LastErrorCode <> _LE_NO_ERROR then
    'Дата задана невенно
    ...
End If
```

1C

```
ДатаВремяНачалаВыборки = СоединениеЛогФайлом.BeginDateText;
```

```
Сообщить("Начальная дата/время выборки: " + ДатаВремяНачалаВыборки);

// Задаем другую дату
СоединениеЛогФайлом.BeginDateText = "2023-07-10 15:00:00";
```

C++

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

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrBeginDateText;
hr = l_spIDataRequest->get_BeginDateText(&l_bstrBeginDateText);
// Задаем другую дату
hr = l_spIDataRequest->put_BeginDateText(CComBSTR(_T("2023-07-10
15:00:00")));
if (hr != S_OK) {
    // Дата задана неверно
}
...
```

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**) может принимать значения:

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

VBScript

```
Dim l_sEndDateText
' Получаем дату/время конца диапазона выборки данных
l_sEndDateText = l_objDataRequest.EndDateText
' Задаем другую дату
l_objDataRequest.EndDateText = "2023-07-10 16:00:00"
```

1C

```
ДатаВремяКонцаВыборки = СоединениеЛогФайлом.EndDateText;

Сообщить("Начальная дата/время выборки: " + ДатаВремяКонцаВыборки);

// Задаем другую дату
```

```
СоединениеЛогФайлом.EndDateText = "2023-07-10 16:00:00";
```

C++

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

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrEndDateText;
hr = l_spIDataRequest->get_EndDateText(&l_bstrEndDateText);
// Задаем другую дату
hr = l_spIDataRequest->put_EndDateText(CComBSTR(_T("2023-07-10 16:00:00")));
...
```

Panel

Получает или задает шаблон имени пульта, на котором была нажата кнопка ответа. Имя пульта может быть обезличенным, например “Окно 1” или иметь имя сотрудника, работающего за этим пультом, например “Иванова Мария”.
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, с именем пульта удовлетворяющем шаблону. В шаблоне имени можно задавать специальные символы:

***** (звездочка) - любой текст в данном месте текста. Например, при задании шаблона имени пульта “Иванов*”, будут выбраны все записи, в которых в имя пульта начинается с Иванов. Этому условию соответствуют, например, “Иванов Петр”, “Иванова Мария*”, “Иванов” и т.п.

? (знак вопроса) - любой одиночный символ в данном месте текста. Например, шаблону “Окно ?” будут соответствовать имена пультов “Окно 1”, “Окно 2”, “Окно 9”, но не соответствовать “Окно 10”.

VBScript

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

1C

```
// получаем шаблон имени пульта
ШаблонИмениПульта = СоединениеЛогФайлом.Panel;

// Задаем свой шаблон имени пульта
СоединениеЛогФайлом.Panel = "Окно 1";
```

C++

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

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrPanel;
hr = l_spIDataRequest->get_Panel(&l_bstrPanel);
// Задаем другое значение
hr = l_spIDataRequest->put_Panel(CComBSTR(_T("Иванов*")));
...
```

AnswerAlias

Получает или задает шаблон псевдонима варианта ответа вопроса, который дал клиент. Псевдомим варианта ответа и текст ответа это разные вещи, и как правило, они отличаются. Текст варианта ответа в записях LOG файле отсутствует. Вместо него, каждому варианту ответа на вопрос соответствует некоторый псевдоним. Как правило (но не обязательно), псевдонимы ответа имеют значения строк с номерами, например:

“1” – Отлично

“2” – Плохо

“3” – Так себе…

Если в LOG файле фиксируются ответы на разные вопросы, то псевдонимы вариантов ответов должны быть различными для разных вопросов. Например “1”, “2” и “3” для первого вопроса, “11”, “12”, “13”, “14” и “15” для второго вопроса и т.д. Можно также придумать псевдонимы, совмещающие принадлежность к вопросу с вариантом ответа, например, “Q1:A1”, “Q1:A2”… “Q2:A1”, “Q2:A2” и т.д.

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, с псевдонимами вариантов ответов, соответствующих шаблону. В шаблоне можно задавать специальные символы:

***** (звездочка) - любой текст в данном месте текста. Например, при задании шаблона “Q1:A*”, будут выбраны все записи, в которых в псевдонимя вариантов ответа начинаются с Q1:, т.е. относятся к первому вопросу.

? (знак вопроса) - любой одиночный символ в данном месте текста. Например, шаблону “?” будут соответствовать псевдонимы вариантов ответа “ 1”, “ 2” и “3”, но не соответствовать “11”, “12” и “13”.

VBScript

```
Dim l_sAnswerAlias
' Получаем текущее значение
l_sAnswerAlias = l_objDataRequest.AnswerAlias
' Задаем другое значение
l_objDataRequest.AnswerAlias = "1"  'Положительная оценка
```

1C

```
// получаем шаблон псевдонима варианта ответа
ШаблонПсевдонимаВариантаОтвета = СоединениеЛогФайлом.AnswerAlias;

// Задаем свой шаблон псевдонима варианта ответа
СоединениеЛогФайлом.AnswerAlias = "1";
```

C++

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

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

```
...
CComPtr<IDataRequest > l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
CComBSTR l_bstrAnswerAlias;
l_spIDataRequest->get_AnswerAlias(&l_bstrAnswerAlias);
// Задаем другое значение
l_spIDataRequest->put_AnswerAlias(CComBSTR(_T("1"))); // Положительная оценка
...
```

AsyncSelectInProgress

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

VBScript

```
...
If l_objDataRequest.AsyncSelect() Then
  While l_objDataRequest. AsyncSelectInProgress
    'Какие либо действия и проверки
  Wend
End If
...
```

1C

```
Если СоединениеЛогФайлом.AsyncSelect() = Истина Тогда
    Пока СоединениеЛогФайлом.AsyncSelectInProgress = Истина Цикл
        // Здесь надо реализовать какой-либо код задержки на короткое время
        // Например, с использованием ПодключитьОбработчикОжидания()
```

```
        ...
        // Можно выполнять другие действия,
        // не относящиеся к объекту СоединениеЛогФайлом
    КонецЦикла;
КонецЕсли;
```

C++

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
// Задали параметры и запускаем асинхронную выборку
VARIANT_BOOL* pbProgress;
l_spIDataRequest->AsyncSelect(&pbProgress);
If (pbProgress) {
    While (pbProgress) {
        Sleep(100);
        l_spIDataRequest->AsyncSelectInProgress(&pbProgress);
    }
    ...
}
...
```

ResultRecordCount

Только для чтения. Число. Возвращает количество записей, выбранных из LOG файла в результате выборки (методом DoSelect() или AsyncSelect()). Если возвращается значение -1, то это означает, что процесс выборки еще не закончен.

VBScript

```
' Задали параметры и выполняем запрос синхронным методом
If l_objDataRequest.DoSelect() Then
  ' Получаем число выбранных записей
  Dim l_nRecordCont
  l_nRecordCont = l_objDataRequest.ResultRecondCount
  ' В цикле запрашиваем записи по индексу
  ...
End If
```

1C

```
// Задали параметры и выполняем запрос синхронным методом
Если СоединениеЛогФайлом.DoSelect() = Истина Тогда
    ЧислоВыбранныхЗаписей = СоединениеЛогФайлом.ResultRecordCount;
    // В цикле запрашиваем результаты выборки по индексу
КонецЕсли;
```

C++

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

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

ResultPanelCount

Только для чтения. Число. Возвращает количество элементов массива неповторяющихся имен пультов, сформированного в результате выборки (методом DoSelect или AsyncSelect). Если возвращается значение -1, то это означает, что процесс выборки еще не закончен. Каждый элемент массива содержит имя пульта и число записей в результирующей выборке, связанных с этим пультом. Имя пульта по его индексу в массиве может быть получено методом GetResultPanelNameByIndex(), а число записей методом GetResultPanelCounterByIndex().

VBScript

```
' Задали параметры и выполняем запрос синхронным методом
If l_objDataRequest.DoSelect() Then
  ' Получаем число различных имен пультов в выборке
  Dim l_nResultPanelCount
  l_nResultPanelCount = l_objDataRequest.ResultPanelCount
  If l_nResultPanelCount <> -1 Then
    ' В цикле запрашиваем информацию о каждом пульте по индексу
    ...
  End If
End If
```

1C

```
// Задали параметры и выполняем запрос синхронным методом
Если СоединениеЛогФайлом.DoSelect() = Истина Тогда
    ЧислоПультов = СоединениеЛогФайлом.ResultPanelCount;
    Если ЧислоПультов <> -1 Тогда
      // В цикле запрашиваем информацию о каждом пульте по индексу
    ...
    КонецЕсли;
КонецЕсли;
```

C++

```
HRESULT CDataRequest::get_ResultPanelCount([out, retval] LONG* pValue);
```

```
...
HRESULT hr = l_spIDataRequest.DoSelect();
if (hr == S_OK) {
  LONG l_PanelCount;
  l_spIDataRequest->get_ResultPanelCount(&l_PanelCount);
```

```
  if (l_PanelCount != -1) {
    // В цикле запрашиваем информацию о каждом пульте по индексу
    ...
  }
}
...
```

ResultAnswerAliasCount

Только для чтения. Число. Возвращает количество элементов массива неповторяющихся псевдонимов вариантов ответа, сформированного в результате выборки (методом **DoSelect** или **AsyncSelect**). Если возвращается значение -1, то это означает, что процесс выборки еще не закончен. Каждый элемент массива содержит псевдоним варианта ответа и число записей в результирующей выборке, имеющих этот вариант ответа. Псевдоним вариантаответв по его индексу в массиве может быть получено методом **GetResultAnswerAliasNameByIndex()**, а число записей методом **GetResultAnswerAliasCounterByIndex()**.

VBScript

```
' Задали параметры и выполняем запрос синхронным методом
If l_objDataRequest.DoSelect() Then
  ' Получаем число различных псевдонимв вариантов ответа в выборке
  Dim l_nResultAnserAliasCount
  l_nResultAnserAliasCount = l_objDataRequest. ResultAlswerAliasCount
  If l_nResultAnserAliasCount <> -1 Then
    ' В цикле запрашиваем информацию о каждом псевдониме варианта ответа
    ...
  End If
End If
```

1C

```
// Задали параметры и выполняем запрос синхронным методом
Если СоединениеЛогФайлом.DoSelect() = Истина Тогда
    ЧислоВариантовОтвета = СоединениеЛогФайлом. ResultAlswerAliasCount;
    Если ЧислоВариантовОтвета <> -1 Тогда
      // В цикле запрашиваем информацию о каждом псевдониме варианта ответа
    ...
    КонецЕсли;
КонецЕсли;
```

C++

```
HRESULT CDataRequest::get_ ResultAlswerAliasCount
                ([out, retval] LONG* pValue);
```

```
...
HRESULT hr = l_spIDataRequest.DoSelect();
if (hr == S_OK) {
  LONG l_AlswerAliasCount;
  l_spIDataRequest->get_ ResultAlswerAliasCount(&l_AlswerAliasCount);
  if (l_AlswerAliasCount != -1) {
    // В цикле запрашиваем информацию о каждом псевдониме варианта ответа
    ...
```

```
    }
}
...
```

Методы объекта DataRequest

Методы (Methods) объекта можно разделить на следующие категории:

- Методы очистки ошибки последней операции **ClearLastError**, сброса параметров выборки **ResetRequestParameters** и метод задания максимального диапазона дат выборки **SetDateRangeMax**;
- 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;
- Методы, относящиеся к получению результатов выборки:
    - Метод, возвращающий объект **ResultRecord** из набора выбранных записей;
    - Метод **GetResultPanelNameByIndex**, возвращающий имя пульта из коллекции имен пультов в результате выборки, по его индексу;
    - Метод **GetResultPanelCounterByIndex**, возвращающий число выбранных записей, относящихся к пульту по его индексу;
    - Метод **GetResultAnswerAliasNameByIndex**, возвращающий имя псевдонима варианта ответа из коллекции имен псевдонимов вариантов ответа в результате выборки, по его индексу;
    - Метод **GetResultAnswerAliasCounterByIndex**, возвращающий число выбранных записей, относящихся к псевдониму варианта ответа по его индексу;

ClearLastError

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

VBScript

```
l_objDataRequest.ClearLastError
```

1C

```
СоединениеЛогФайлом.ClearLastError()
```

C++

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

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
l_spIDataRequest->ClearLastError();
...
```

ResetRequestParameters

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

- Свойство **LogFileFolder 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;
- Свойства Panel и AnswerAlias устанавливаются в пустую строку

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

VBScript

```
l_objDataRequest.BeginDateText = "2023-07-10 15:00:00"
l_objDataRequest.EndDateText = "2023-07-10 16:00:00"
l_objDataRequest.Panel = "Окно 1"
l_objDataRequest.AnswerAlias = "1"

...

' Выполняем запрос
l_objDataRequest.DoSelect()
...
' Получаем результат выборки
...

' Сбрасываем значения свойств
l_objDataRequest.ResetRequestParameters()
...

' Повторно используем объект
```

1C

```
СоединениеЛогФайлом.BeginDateText = "2023-07-10 15:00:00";
СоединениеЛогФайлом.EndDateText = "2023-07-10 16:00:00";
СоединениеЛогФайлом.Panel = "Окно 1";
СоединениеЛогФайлом.AnswerAlias = "1";

// Задали параметры и выполняем запрос синхронным методом
СоединениеЛогФайлом.DoSelect();

// Получаем результат выборки
...

// Сбрасываем значения свойств
СоединениеЛогФайлом.ResetRequestParameters();
```

```
...

// Повторно используем объект
```

C++

```
HRESULT CDataRequest::ResetRequestParameters(void);
```

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
l_spIDataRequest->BeginDateText(CComBSTR(_T("2023-07-10 15:00:00")));
l_spIDataRequest->EndDateText(CComBSTR(_T("2023-07-10 16:00:00")));
l_spIDataRequest->put_Panel(CComBSTR(_T("Окно 1")));
l_spIDataRequest->put_AnswerAlias(CComBSTR(_T("1")));
...
// Выполняем запрос
l_spIDataRequest->DoSelect();
...
// Получаем результат выборки
...

// Сбрасываем значения свойств
l_spIDataRequest->ResetRequestParameters();
...
// Повторно используем объект
```

SetDateRangeMax

Задает максимальный диапазон дат выборки. Начальная дата выборки устанавливается в 0.0 (30 декабря 1899 года), а конечная дата в + 1 год от текущей даты.

Не имеет параметров и не возвращает никаких значений.

VBScript

```
l_objDataRequest.BeginDateText = "2023-07-10 15:00:00"
l_objDataRequest.EndDateText = "2023-07-10 16:00:00"

...

' Выполняем запрос
l_objDataRequest.DoSelect()
...
' Получаем результат выборки
...

' Задаем максимальный диапазон дат выборки
l_objDataRequest.DatDateRangeMax
...

' Повторно выполняем выборку с этим диапазоном дат
```

1C

```
СоединениеЛогФайлом.BeginDateText = "2023-07-10 15:00:00";
СоединениеЛогФайлом.EndDateText = "2023-07-10 16:00:00";

// Задали параметры и выполняем запрос синхронным методом
СоединениеЛогФайлом.DoSelect();

// Получаем результат выборки
...

// Задаем максимальный диапазон дат выборки
СоединениеЛогФайлом.SetDateRangeMax();
...

// Повторно выполняем выборку с этим диапазоном дат
```

C++

```
HRESULT CDataRequest::SetDateRangeMax (void);
```

```
...
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);
...
l_spIDataRequest->BeginDateText(CComBSTR(_T("2023-07-10 15:00:00")));
l_spIDataRequest->EndDateText(CComBSTR(_T("2023-07-10 16:00:00")));
...
// Выполняем запрос
l_spIDataRequest->DoSelect();
...
// Получаем результат выборки
...

// Задаем максимальный диапазон дат выборки
l_spIDataRequest->SetDateRangeMax ();
...
// Повторно выполняем выборку с этим диапазоном дат
```

DoSelect

Выполняет синхронный запрос к LOG файлу, находящемуся в заданной папке и параметров выборки:

- Интервала дат/времен выборки;
- Шаблона имени пульта (**Panel**), если задано;
- Шаблона псевдонима варианта ответа (**AnswerAlias**), если задан;

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, что не является стандартным приемом для синхронной выборки.

Метод производит следующие действия:

- Открывает и читает содержимое LOG файла (включая файл ротации);
- Сохраняет результаты выборки (с учетом параметров выборки) во внутренних объектах;

- Закрывает LOG файл;
- Возвращает управление вызывающей программе и сообщает информацию об успехе или ошибке в процессе выполнения.

Метод возвращает TRUE, если выборка данных была 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_LOG_FILE_FOLDER_NOT_SET - "Не задана папка LOG файла ".
- _LE_LOG_FILE_FOLDER_NOT_EXIST - "Папка LOG файла не существует или недоступна".
- _LE_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется". Ранее был запущен процесс асинхронной выборки (метод **AsyncSelect**), который на данный момент еще не завершен.
- _LE_OPEN_LOG_FILE_ERROR – Ошибка открытия LOG файла. Используйте свойство **LastErrorText** для получения детального описания причины ошибки.
- _LE_LOG_FILE_WRONG_FORMAT - "Файл имеет некорректный формат записи". Возможно, Log-файл использует текстовый формат записей. Требуется формат CSV. Формат LOG файла задается в настройках программы EPM-Agent Plus.
- _LE_READ_LOG_FILE_ERROR – Ошибка чтения LOG файла. Используйте свойство **LastErrorText** для получения детального описания причины ошибки.
- _LE_OUT_OF_MEMORY - "Недостаточно свободной памяти в системе ".
- _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_objDataRequest
'Создаем объект
Set l_objDataRequest = CreateObject("PLLogConnector.DataRequest")

'Задаем значения параметров для выборки
...

'Выполняем запрос синхронно
Dim l_Success
l_Success = l_objDataRequest.DoSelect()
```

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

1C

```
// Создать объект
СоединениеLogФайлом = Новый COMобъект("PLLogConnector64.DataRequest.1");

// Задаем значения параметров для выборки
...

// Выполняем запрос синхронно
УспехВыборки = СоединениеLogФайлом.DoSelect();

Если УспехВыборки = Истина Тогда
  // Получаем результат выборки
  ...
Иначе
  Сообщить("Ошибка выполнения запроса (" +
      СоединениеLogФайлом.LastErrorCode() + "):" +
      СоединениеLogФайлом.LastErrorText());
  ...
КонецЕсли;
```

C++

```
HRESULT CDataRequest::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 "PLLogConnector_i.c" // для 32-х разрядной версии
#include "PLLogConnector.h"
...
CoInitialize(NULL);
...
// SMART указатель на объект с интерфейсом IDataRequest
CComPtr<IDataRequest> l_spIDataRequest;
HRESULT hr = l_spIDataRequest.CoCreateInstance(CLSID_DataRequest);

If (hr == S_OK) {
    // Задаем значения нужных параметров для выборки
    ...
```

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

    if (hr != S_OK) {
        CComBSTR l_bstrErrorText;
        l_spIDataRequest->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 DataRequest. 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 DataRequest используется метод SetCallbackNull().

SetCallbackNull

Задает для объекта механизм обратного вызова _CALLBACK_TYPE_NONE.

Метод ничего не возвращает. Тем не менее, после вызова метода «хорошим тоном» будет проверить код последней ошики, т.к. если в момент вызова метода выполняется асинхронная выборка, то механизм обратного вызова изменять нельзя до ее завешения. В этом случае **LastErrorCode** вернет _LE_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".

VBScript

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

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

        EndIf
    EndIf
EndIf
```

1С

```
// Задаем механизм обратного вызова _CALLBACK_TYPE_NONE
СоединениеLogФайлом.SetCallbackNull();

Если СоединениеLogФайлом.LastErrorCode = 0 Тогда
  // Запускаем выборку асинхронно
  Если СоединениеLogФайлом.AsyncSelect() = Истина Тогда
    Цикл
        // Ждем завершения выборки
        // Здесь надо реализовать какой-либо код задержки на короткое время
        // Например, с использованием ПодключитьОбработчикОжидания()
        ...
        // Можно выполнять другие действия,
        // не относящиеся к объекту СоединениеЛогФайлом
    КонецЦикла;

    Если СоединениеLogФайлом.LastErrorCode = 0 Тогда
      // Получаем результат выборки
      ...
    КонецЕсли;
  КонецЕсли;
КонецЕсли;
```

C++

```
HRESULT CDataRequest:: 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_spIDataRequest.SetCallbackNull();

if (hr == S_OK) {
    // Запускаем выборку асинхронно
```

```
    VARIANT_BOOL bInProgress;
    hr = spIDataRequest-> AsyncSelect(&bInProgress);
    if (hr == S_OK) {
        time_t tmStartTime, tmCurTime;
        time(&tmStartTime);

        // Ждем завершения выборки, но не более 5 секунд
        bInProgress = TRUE;
        BOOL bInterrupted = FALSE;
        do {
            spIDataRequest->get_AsyncSelectInProgress(&bInProgress);
            if (bInProgress) {
                time(&tmCurTime);
                if ((tmCurTime - tmStartTime > 5) && !bInterrupted) {
                    // Прерываем процесс выборки
                    spIDataRequest->StopAsyncSelect();
                    bInterrupted = TRUE;
                }
                Sleep(100);
            }
        } while(bInProgress);

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

            if (nLastErrorCode == _LE_NO_ERROR) {
                // Получаем результат выборки
                ...
            }
        }
    }
}
else {
    // выборка выполняется в данный момент…
    ...
}
...
```

_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 DataConnector дескриптор события. Дескриптор возвращается методом SetCallbackEvent().

SetCallbackEvent

Задает для объекта механизм обратного вызова _CALLBACK_TYPE_EVENT и возвращает дескриптор события. Если в момент вызова метода выполняется асинхронная выборка, то механизм

обратного вызова не будет изменен. В этом случае **LastErrorCode** вернет _LE_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".

VBScript

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

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

1C

```
// Задаем механизм обратного вызова _CALLBACK_TYPE_EVENT
СобытиеОкончанияВыборки = СоединениеLogФайлом.SetCallbackEvent();

Если СоединениеLogФайлом.LastErrorCode = 0 Тогда
  // Запускаем выборку асинхронно
Иначе
  // Дыборка выполняется в данный момент и механизм обратного вызова изменять
нельзя
КонецЕсли;
```

C++

```
HRESULT CDataRequest:: 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_spIDataRequest.SetCallbackEvent(&hEvent);

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

```
...
```

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 CDataRequest::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_objDataRequest.SetCallbackEvent

If l_objDataRequest.LastErrorCode = 0 Then
    'Запускаем выборку асинхронно
```

```
    If l_objDataRequest.AsyncSelect Then
        Dim nRetCode

        Do
            ' Ждем завершения процесса выборки. 2 секунды
            nRetCode = objDataRequest.WaitForCallbackEvent(hEvent, 2000)
            If nRetCode = 0 Then
                Exit Do
            End If
            ' Иначе выполняем какие-либо действия и ждем дальше
            ...
        Loop
        If l_objDataRequest.LastErrorCode = 0 Then
            'Получаем результат выборки
            ...
        EndIf
    EndIf
Else
    ' выборка выполняется в данный момент…
EndIf
```

1C

```
// Задаем механизм обратного вызова _CALLBACK_TYPE_EVENT
СобытиеОкончанияВыборки = СоединениеLogФайлом.SetCallbackEvent();

Если СоединениеLogФайлом.LastErrorCode = 0 Тогда
  // Запускаем выборку асинхронно
  Если СоединениеLogФайлом.AsyncSelect() = Истина Тогда
    Пока СоединениеLogФайлом.WaitForCallbackEvent(СобытиеОкончанияВыборки,
2000) <> 0 Цикл
      // Выборка еще выполняется. Можем что-нибудь делать и жать дальше
      // Либо прервать выборку, вызвав метод StopAsyncSelect()
      ...
    Конец цикла
  КонецЕсли;

КонецЕсли;
```

_CALLBACK_TYPE_WINDOW

Механизм обратного вызова через посылку сообщения окну программы. По окончании асинхронной выборки, в заданное окно посылается сообщение. Приложение, получив сообщение должно проверить код последней ошибки и перейти к получению результата выборки. Для использования механизма _CALLBACK_TYPE_WINDOW, до начала асинхронной выборки необходимо вызвать метод SetCallbackWindow().

SetCallbackWindow

```
HRESULT CDataRequest::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 DataRequest, 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").
При получении оконной функцией сообщения **uMsg**, в параметре **lParam**, будет содержаться указатель на интерфейс объекта.

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

C++

```
void SomeFunc() {
  ...
  l_spIDataRequest->SetCallbackWindow(g_hwndMain, WM_USER+200);

  BOOL bInProgress;
  HRESULT hr = l_spIDataRequest->AsyncSelect(&bInProgress);
  ...
}

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

        case WM_USER+200:
        {
            IDataRequest *l_pIDataRequest = (IDataRequest *)lParam;
            BOOL bInProgress = TRUE;

            LONG l_lLastErrorCode;
             l_pIDataRequest->get_LastErrorCode(&l_lLastErrorCode);
             If (lLastErrorCode == _LE_NO_ERROR) {
                // Не было ошибки или прерывания 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 TRUE, если асинхронной процесс выборки успешно начат, и FALSE в противном случае.

В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства **LastErrorCode**, а тест ошибки - **LastErrorText**. Возможны следующие ошибки:

- **_LE_SELECTION_IN_PROGRESS** (4) - "Асинхронная выборка еще выполняется"
- Другой код ошибки, отличный от 0 - "Ошибка создания потока: …"

VBScript

```
...
'Задаем параметры выборки и механизм обратного вызова
...
'Запускаем выборку асинхронно
If l_objDataRequest.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_objDataRequest.LastErrorCode
    'Получить текст ошибки
    Dim sErrorText
    sErrorText = l_objDataRequest.LastErrorText
    ...
EndIf
```

1C

```
// Задаем параметры выборки и механизм обратного вызова
СобытиеОкончанияВыборки = СоединениеLogФайлом.SetCallbackEvent();
...
Если СоединениеLogФайлом.AsyncSelect() = Истина Тогда
    // Асинхронная выборка в процессе работы
    // Ожидать завершения, в зависимости от механизма обратного вызова
Иначе
    // Ошибка запуска асинхронной выбрки. Получить код ошибки
    КодПоследнейОшибки = СоединениеLogФайлом.LastErrorCode;
    // Получить текст ошибки
    ТекстПоследнейОшибки = СоединениеLogФайлом.LastErrorText;
    ...
КонецЕсли;
```

C++

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

```
...
// Задаем параметры выборки и механизм обратного вызова
...
BOOL bInProgress;
HRESULT hr = l_spIDataRequest->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_spIDataRequest->get_LastErrorCode(&l_lLastErrorCode);
     // Получить текст ошибки
     CComBSTR l_bstrErrorText;
     l_spIDataRequest->get_LastErrorText(&l_bstrErrorText);
    ...
}
...
```

StopAsyncSelect

Дает сигнал (взводит внутреннее событие) на останов процесса асинхронной выборки данных. Процесс выборки не будет прерван моментально, поэтому приложение по-прежнему должно ожидать завершения процесса выборки. Если процесс асинхронной выборки будет остановлен по событию останова, то код последней ошибки примет значение:

_LE_SELECTION_INTERRUPTED_BY_USER (8) - "Процесс выборки прерван пользователем"

Метод не возвращает никаких значений.

VBScript

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

        'Ждем завершения выборки, но не более 5 секунд
        While l_objDataRequest.AsyncSelectInProgress
            If Not bInterrupted And (DateDiff("s", Now, dtStart) > 5) Then
                l_objDataRequest.StopAsyncSelect()
                bInterrupted = True
            End If
        Wend

        Select Case l_objDataRequest.LastErrorCode
            Case _LE_SELECTION_INTERRUPTED_BY_USER
                'Было прервано по StopAsyncSelect
                ...
            Case 0
                'Получаем результат выборки
                ...
            Case Else
```

```
                'Другая ошибка в процессе выборки
                'Получить код ошибки
                Dim nErrorCode
                nErrorCode = l_objDataRequest.LastErrorCode
                'Получить текст ошибки
                Dim sErrorText
                sErrorText = l_objDataRequest.LastErrorText
                ...

            ...
        End Select
    EndIf
EndIf
```

1С

```
// Задаем параметры выборки
// Механизм обратного вызова _CALLBACK_TYPE_NONE
СобытиеОкончанияВыборки = СоединениеLogФайлом.SetCallbackNone();
...
Если СоединениеLogФайлом.AsyncSelect() = Истина Тогда
    // Асинхронная выборка в процессе работы
    // Ждем завершения выборки, но не более 5 секунд
    МаксЗадержкаСек = 5;
    КонДата = ТекущаяДата() + МаксЗадержкаСек;

     Пока ТекущаяДата() < КонДата И
          СоединениеLogФайлом.AsyncSelectInProgress = Истина Цикл
       // ждем....
     КонецЦикла;

     Если СоединениеLogФайлом.AsyncSelectInProgress = Истина Тогда
        СоединениеLogФайлом.StopAsyncSelect();
     КонецЕсли;
     Пока СоединениеLogФайлом.AsyncSelectInProgress = Истина Цикл
       // ждем....
     КонецЦикла;

    КодПоследнейОшибки = СоединениеLogФайлом.LastErrorCode;

    Если КодПоследнейОшибки = 0 Тогда
        // Получаем результат выборки
        ...
    ИначеЕсли КодПоследнейОшибки = 8 Тогда
        // Было прервано по StopAsyncSelect
        ...
    Иначе
        // Другая ошибка в процессе выборки
        // Получить код ошибки
        КодПоследнейОшибки = СоединениеLogФайлом.LastErrorCode;
        // Получить текст ошибки
        ТекстПоследнейОшибки = СоединениеLogФайлом.LastErrorText;
        ...
    КонецЕсли;
    ...
КонецЕсли;
```

C++

```
HRESULT CDataRequest::StopAsyncSelect();
```

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

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

        // Ждем завершения выборки
        // Если выборка затянется на 5 секунд, то прерываем ее
        bInProgress = TRUE;
        BOOL bInterrupted = FALSE;
        do {
            spIDataRequest->get_AsyncSelectInProgress(&bInProgress);
            if (bInProgress) {
                time(&tmCurTime);
                if ((tmCurTime - tmStartTime > 5) && !bInterrupted) {
                    // Прерываем процесс выборки
                    spIDataRequest->StopAsyncSelect();
                    bInterrupted = TRUE;
                }
                Sleep(100);
            }
        } while(bInProgress);

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

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

Получение результатов выборки

В результате успешной выборки, вне зависимости синхронной или асинхронной, объект DataRequest сохраняет во внутренних структурах результы выборки (записи). Кроме того, формируется две коллекции:

- Список имен пультов с числом записей в результате выборки по каждому пульту;
- Список псевдонимов вариантов ответа с числом записей в результате выборки по каждому варианту ответа;

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

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

В случае ошибки, код ошибки устанавливается в:

- _LE_ASYNC_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".
- ERROR_INVALID_PARAMETER (87) - "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 приложения **не создаются напрямую**. Вместо этого, у объекта DataRequest запрашивается элемент массива выбранных записей. Полученный от метода объект (интерфейс объекта) **должен быть удален в коде приложения**.

VBScript

```
'Выполняем запрос
l_objDataRequest.DoSelect

If l_objDataRequest.LastErrorCode = 0 Then
    'Запрашиваем число выбранных записей
    Dim l_nRecordCount : l_nRecordCount = l_objDataRequest.ResultRecordCount
    If l_nRecordCount > 0 Then
        Dim l_objCurrentRecord
        'Запрашиваем объект с индексом 0
        Set l_objCurrentRecord = l_objDataRequest.GetResultRecord(0)
        'Запрашиваем свойства объекта ResultRecord
        Dim l_dt : l_dt = l_objCurrentRecord.DateTime
        ...
```

```
        'Удаляем объект (необязательно для скриптовых языков)
        Set l_objCurrentRecord = Nothing
    End If
End If
...
```

1C

```
// Выполняем запрос синхронно
Если СоединениеLogФайлом.DoSelect() = Истина Тогда
    ЧислоВыбранныхЗаписей = СоединениеLogФайлом.ResultRecordCount;
    Если ЧислоВыбранныхЗаписей > 0 Тогда
        // Запрашиваем объект с индексом 0
        ЗаписьLogФайла = СоединениеLogФайлом.GetResultRecord(0);
        // Запрашиваем свойства объекта ResultRecord
        ДатаВремяСобытия = ЗаписьLogФайла.DateTimeText
        ...
    КонецЕсли;
КонецЕсли;
```

C++

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

if (hr == S_OK) {
    // Запрашиваем число элементов массива выбранных записей
    LONG l_lRecordCount = 0;
    l_spIDataRequest->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_spIDataRequest->GetResultRecord(
                                    i, &l_spCurrentRecord))
            {
                // Запрашиваем свойства объекта CurrentRecord
                DATE l_dt;
                l_spCurrentRecord->get_DateTime(&l_dt);
                ...
                // Удаляем объект (необязательно для Smart pointer)
                l_spCurrentRecord = NULL;
            }
        }   // for
    }
}
...
```

GetResultPanelNameByIndex

```
HRESULT CDataRequest::GetResultPanelNameByIndex([in] LONG lIndex,
    [out, retval] BSTR* pName);
```

Возвращает имя пульта из массива неповторяющихся имен пультов, полученного в результате выборки, по его индексу. Строка. Количество элементов в массиве имен пультов может быть

получено из свойства **ResultPanelCount**. В случае ошибки, возвращается пустая строка, а код последней ошибки устанавливается в:

- _LE_ASYNC_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".
- ERROR_INVALID_PARAMETER (87) - "Неверно задан индекс массива".

VBScript

```
'Выполняем запрос
l_objDataRequest.DoSelect

If l_objDataRequest.LastErrorCode = 0 Then
    'Запрашиваем число неповторяющихся имен пультов
    Dim l_nUniquePanelCount
    l_nUniquePanelCount = l_objDataRequest.ResultPanelCount
    If l_nUniquePanelCount > 0 Then
        Dim i
        For i = 0 To l_nUniquePanelCount - 1
          Dim l_sUniquePanelName
          'Запрашиваем имя пульта с индексом i
          l_sUniquePanelName = l_objDataRequest. GetResultPanelNameByIndex(i)
          ...
        Next
    End If
End If
...
```

1C

```
// Выполняем запрос синхронно
Если СоединениеLogФайлом.DoSelect() = Истина Тогда
    // Запрашиваем число неповторяющихся имен пультов
    ЧислоИменПультов = СоединениеLogФайлом.ResultPanelCount;
    Если ЧислоИменПультов > 0 Тогда
        // Запрашиваем имя пульта с индексом 0
        ИмяПульта = СоединениеLogФайлом.GetResultPanelNameByIndex(0);
        ...
    КонецЕсли;
КонецЕсли;
```

C++

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

if (hr == S_OK) {
    // Запрашиваем число неповторяющихся имен пультов
    LONG l_UniquePanelCount = 0;
    l_spIDataRequest->get_ResultRecordCount(&l_UniquePanelCount);

    if (l_UniquePanelCount > 0) {
        for (int i=0; i < l_UniquePanelCount; i++) {
            CComBSTR l_bstrPanelName;
            // Запрашиваем имя пульта с индексом i
```

```
            l_spIDataRequest->GetResultPanelNameByIndex(
                                    i, &l_bstrPanelName);
            ...
        }   // for
    }
}
...
```

GetResultPanelCounterByIndex

```
HRESULT CDataRequest:: GetResultPanelCounterByIndex([in] LONG lIndex,
    [out, retval] LONG* pValue);
```

Возвращает количество записей в результате выборки, связанное с данным пультом, по его индексу. Целое число. Количество элементов в массиве пультов может быть получено из свойства **ResultPanelCount**. В случае ошибки, возвращается значение -1, а код последней ошибки устанавливается в:

- _LE_ASYNC_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".
- ERROR_INVALID_PARAMETER (87) - "Неверно задан индекс массива".

VBScript

```
'Выполняем запрос
l_objDataRequest.DoSelect

If l_objDataRequest.LastErrorCode = 0 Then
    'Запрашиваем число неповторяющихся имен пультов
    Dim l_nUniquePanelCount
    l_nUniquePanelCount = l_objDataRequest.ResultPanelCount
    If l_nUniquePanelCount > 0 Then
        Dim i
        For i = 0 To l_nUniquePanelCount - 1
          Dim l_sUniquePanelName
          'Запрашиваем имя пульта с индексом i
          l_sUniquePanelName = l_objDataRequest. GetResultPanelNameByIndex(i)
          Dim l_nUniquePanelCounter
          'Запрашиваем число записей в результате выборки
          ' связанное с этим пультом
          l_nUniquePanelCounter = \
               l_objDataRequest.GetResultPanelCounterByIndex(i)
          ...
        Next
    End If
End If
...
```

1C

```
// Выполняем запрос синхронно
Если СоединениеLogФайлом.DoSelect() = Истина Тогда
    // Запрашиваем число неповторяющихся имен пультов
    ЧислоИменПультов = СоединениеLogФайлом.ResultPanelCount;
    Если ЧислоИменПультов > 0 Тогда
        // Запрашиваем имя пульта с индексом 0
```

```
        ИмяПульта = СоединениеLogФайлом.GetResultPanelNameByIndex(0);
        // Запрашиваем число записей в результате выборки
        // связанное с этим пультом
        ЧислозаписейДляПульта =
            СоединениеLogФайлом.GetResultPanelCounterByIndex(0);
        ...
    КонецЕсли;
КонецЕсли;
```

C++

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

if (hr == S_OK) {
    // Запрашиваем число неповторяющихся имен пультов
    LONG l_UniquePanelCount = 0;
    l_spIDataRequest->get_ResultRecordCount(&l_UniquePanelCount);

    if (l_UniquePanelCount > 0) {
        for (int i=0; i < l_UniquePanelCount; i++) {
            CComBSTR l_bstrPanelName;
            // Запрашиваем имя пульта с индексом i
            l_spIDataRequest->GetResultPanelNameByIndex(
                                i, &l_bstrPanelName);
           LONG l_PanelRecordCount = 0;
            l_spIDataRequest->GetResultPanelCounterByIndex(
                                i, & l_PanelRecordCount);
            ...
        }   // for
    }
}
```

GetResultAnswerAliasNameByIndex

```
HRESULT CDataRequest:: GetResultAnswerAliasNameByIndex([in] LONG lIndex,
    [out, retval] BSTR* pAnswerAlias);
```

Возвращает псевдоним варианта ответа из массива неповторяющихся псевдонимов вариантов ответа, полученного в результате выборки, по его индексу. Строка. Количество элементов в массиве псевдонимов вариантов ответа может быть получено из свойства **ResultAnswerAliasCount**. В случае ошибки, возвращается пустая строка, а код последней ошибки устанавливается в:

- _LE_ASYNC_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".
- ERROR_INVALID_PARAMETER (87) - "Неверно задан индекс массива".

VBScript

```
'Выполняем запрос
l_objDataRequest.DoSelect

If l_objDataRequest.LastErrorCode = 0 Then
```

```
    'Запрашиваем число неповторяющихся псевдонимов вариантов ответов
    Dim l_nUniqueAnswerAliasCount
    l_nUniqueAnswerAliasCount = l_objDataRequest.ResultAnswerAliasCount
    If l_nUniqueAnswerAliasCount > 0 Then
        Dim i
        For i = 0 To l_nUniqueAnswerAliasCount - 1
          Dim l_sUniqueAnswerAlias
          'Запрашиваем псевдоним варианта ответа с индексом i
          l_sUniqueAnswerAlias = \
               l_objDataRequest. GetResultAnswerAliasNameByIndex(i)
          ...
        Next
    End If
End If
...
```

1C

```
// Выполняем запрос синхронно
Если СоединениеLogФайлом.DoSelect() = Истина Тогда
    // Запрашиваем число неповторяющихся псевдоним варианта ответа
    ЧислоПсевдонимовВариантовОтветов =
           СоединениеLogФайлом.ResultAnswerAliasCount;
    Если ЧислоПсевдонимовВариантовОтветов > 0 Тогда
        // Запрашиваем псевдоним варианта ответа с индексом 0
        ИмяПсевдонимаВариантаОтвета =
            СоединениеLogФайлом.GetResultAnswerAliasNameByIndex(0);
        ...
    КонецЕсли;
КонецЕсли;
```

C++

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

if (hr == S_OK) {
    // Запрашиваем число неповторяющихся псевдонимов варианта ответа
    LONG l_UniqueAnswerAliasCount = 0;
    l_spIDataRequest->get_ResultAnswerAliasCount(&l_UniqueAnswerAliasCount);

    if (l_UniqueAnswerAliasCount > 0) {
        for (int i=0; i < l_ l_UniqueAnswerAliasCount; i++) {
            CComBSTR l_bstrAnswerAliasName;
            // Запрашиваем псевдоним варианта ответа с индексом i
            l_spIDataRequest->GetResultAnswerAliasNameByIndex(
                                    i, & l_bstrAnswerAliasName);
            ...
        }   // for
    }
}
...
```

GetResultAnswerAliasCounterByIndex

```
HRESULT CDataRequest:: GetResultAnswerAliasCounterByIndex([in] LONG lIndex,
    [out, retval] LONG* pValue);
```

Возвращает количество записей в результате выборки, связанное с данным псевдонимом варианта ответа, по его индексу. Целое число. Количество элементов в массиве псевдонимов варианта ответа может быть получено из свойства **ResultAnswerAliasCount**. В случае ошибки, возвращается значение -1, а код последней ошибки устанавливается в:

- _LE_ASYNC_SELECTION_IN_PROGRESS - "Асинхронная выборка еще выполняется".
- ERROR_INVALID_PARAMETER (87) - "Неверно задан индекс массива".

VBScript

```
'Выполняем запрос
l_objDataRequest.DoSelect

If l_objDataRequest.LastErrorCode = 0 Then
    'Запрашиваем число неповторяющихся псевдонимов вариантов ответа
    Dim l_nUniqueAnswerAliasCount
    l_nUniqueAnswerAliasCount = l_objDataRequest.ResultAnswerAliasCount
    If l_nUniqueAnswerAliasCount > 0 Then
        Dim i
        For i = 0 To l_nUniqueAnswerAliasCount - 1
          Dim l_sUniqueAnswerAliasName
          'Запрашиваем имя псевдонима варианта ответа с индексом i
          l_sUniqueAnswerAliasName = \
               l_objDataRequest. GetResultAnswerAliasNameByIndex(i)
          Dim l_nUniqueAnswerAliasCounter
          'Запрашиваем число записей в результате выборки
          ' связанное с этим псевдонимом варианта ответа
          l_nUniqueAnswerAliasCounter = \
               l_objDataRequest.GetResultAnswerAliasCounterByIndex(i)
          ...
        Next
    End If
End If
...
```

1C

```
// Выполняем запрос синхронно
Если СоединениеLogФайлом.DoSelect() = Истина Тогда
    // Запрашиваем число неповторяющихся псевдонимов вариантов ответа
    ЧислоПсевдонимовВариантовОтветов =
           СоединениеLogФайлом.ResultAnswerAliasCount;
    Если ЧислоПсевдонимовВариантовОтветов > 0 Тогда
        // Запрашиваем псевдоним варианта ответа с индексом 0
        // Запрашиваем число записей в результате выборки
        // связанное с этим псевдонимом варианта ответа
        ЧислозаписейДляПсевдонимаВариантаОтвета =
            СоединениеLogФайлом.GetResultAnswerAliasCounterByIndex(0);
        ...
    КонецЕсли;
КонецЕсли;
```

C++

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

if (hr == S_OK) {
    // Запрашиваем число псевдонимов вариантов ответа
    LONG l_UniqueAnswerAliasCount = 0;
    l_spIDataRequest->get_ResultRecordCount(&l_UniqueAnswerAliasCount);

    if (l_UniqueAnswerAliasCount > 0) {
        for (int i=0; i < l_UniqueAnswerAliasCount; i++) {
            CComBSTR l_bstrAnswerAliasName;
            // Запрашиваем имя псевдонима варианта ответа с индексом i
            l_spIDataRequest->GetResultAnswerAliasNameByIndex(
                                    i, & l_bstrAnswerAliasName);
           LONG l_AnswerAliasRecordCount = 0;
            l spIDataRequest->GetResultAnswerAliasCounterByIndex(
                                    i, & l_AnswerAliasRecordCount);
            ...
        }   // for
    }
}
```

Объект ResultRecord

После выполнения выборки данных в объекте DataRequest содержится коллекция (массив записей) объектов класса **ResultRecord** с результатами выборки. Объекты **ResultRecord** не создаются напрямую. Они запрашиваются у объекта DataRequest с помощью метода **GetResultRecord()**. Количество элементов в коллекции содержится в свойстве **ResultRecordCount**. Объект имеет набор свойств и 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. В целой части числа содержится число дней от 30 декабря 1899 года. В дробной части содержаться часы, минуты и секунды как доли от суток.

VBScript

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

```
    For i=0 To l_nRecordCount
        Dim l_objCurrentRecord
        'Запрашиваем объект с индексом i
        Set l_objCurrentRecord = l_objDataRequest.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
...
```

1C

Использовать свойство в 1С проблематично, т.к. тип значения «Дата» в 1С не соответствует типу DATE, которое используется в этом свойстве. Используйте свойство DateTimeText.

C++

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

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

if (hr == S_OK) {
    // Запрашиваем число элементов массива выбранных записей
    LONG l_lRecordCount = 0;
    l_spIDataRequest->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_spIDataRequest->GetResultRecord(i, &l_spCurrentRecord))
        {
            // Запрашиваем дату/время события
            DATE l_dt;
            l_spCurrentRecord->get_DateTime(&l_dt);
            ...
            // Удаляем объект. Не обязательно для smart pointer
            l_spCurrentRecord = NULL;
        }
    }
}
......
```

DC

Возвращает имя хоста Концентратора Данных – компьютер от которого информация об ответе была записана в LOG файл. Строка.

VBScript

```
Dim l_objCurrentRecord
'Запрашиваем объект с индексом i
Set l_objCurrentRecord = l_objDataRequest.GetResultRecord(i)
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sDC: l_sDC = l_objCurrentRecord.DC
...
```

1C

```
// Запрашиваем объект с индексом 0
ЗаписьLogФайла = СоединениеLogФайлом.GetResultRecord(0);
DCЗаписи = ЗаписьLogФайла.DC
```

C++

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

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

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

    // Запрашиваем DC
    CComBSTR l_bstrDC;
    l_spCurrentRecord->get_DC(&l_bstrDC);
    ...
    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_objDataRequest.GetResultRecord(0)
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sDC: l_sDC = l_objCurrentRecord.DC
Dim l_sPanel: l_sPanel = l_objCurrentRecord.Panel
...
```

1C

```
// Запрашиваем объект с индексом 0
ЗаписьLogФайла = СоединениеLogФайлом.GetResultRecord(0);
ПультЗаписи = ЗаписьLogФайла.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_spIDataRequest->GetResultRecord(0, &l_spCurrentRecord))
{
    // Запрашиваем дату/время события
    DATE l_dt;
    l_spCurrentRecord->get_DateTime(&l_dt);

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

AnswerAlias

Возвращает псевдоним варианта ответа, который дал клиент. Строка.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDataRequest.GetResultRecord(0)
'Запрашиваем свойства объекта ResultRecord
Dim l_dt : l_dt = l_objCurrentRecord.DateTime
Dim l_sPanel: l_sPanel = l_objCurrentRecord.Panel
Dim l_sAnswerAlias: l_sAnswerAlias = l_objCurrentRecord.AnswerAlias
...
```

1C

```
// Запрашиваем объект с индексом 0
ЗаписьLogФайла = СоединениеLogФайлом.GetResultRecord(0);
ПультЗаписи = ЗаписьLogФайла.Panel
ПсевдонимВариантаОтвета = ЗаписьLogФайла.AnswerAlias
```

C++

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

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

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

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

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

DateTimeText

Возвращает дату время события в виде строки. Формат представлени “ГГГГ-ММ-ДД чч:мм:сс“.

VBScript

```
...
Dim l_objCurrentRecord
'Запрашиваем объект с индексом 0
Set l_objCurrentRecord = l_objDataRequest.GetResultRecord(0)
'Запрашиваем свойства объекта ResultRecord
Dim l_dtText : l_dtText = l_objCurrentRecord.DateTimeText
...
```

1C

```
// Запрашиваем объект с индексом 0
ЗаписьLogФайла = СоединениеLogФайлом.GetResultRecord(0);
ДатаВремяЗаписи = ЗаписьLogФайла.DatetimeText
```

C++

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

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

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

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

    // Запрашиваем дату/время в текстовом формате
    CComBSTR l_bstrDateTimeText;
    l_spCurrentRecord->get_DateTimeText (&l_bstrDateTimeText);
    ...
    l_spCurrentRecord = NULL;
}
...
```

Время жизни модуля и объектов

Модуль COM-компонента (PLLogConnector.dll либо PLLogConnector64.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: PLLogConnector.Request для 32-х разрядного кода и PLLogConnector64.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.