Локальная система распознавания реального времени

BOURNEID Management Component v.2.2.

Руководство разработчика

1. Введение

Документ предназначен для разработчиков программного обеспечения, взаимодействующего с приложением **BourneID** через компонент **BourneIdMgmt**. Такое программное обеспечение (далее **Внешнее ПО** или **приложение**) может:

- Выполнять запросы к программе BourneID и получать в ответ набор данных, характеризующий текущее состояние программы;
- Подписаться на получение событий из программы BourneID. Компонент **BourneIdMgmt** сообщает внешнему ПО о полученном событии. Внешнее ПО запрашивает компонент и получает набор данных события.

BourneIdMgmt - 32-х разрядный COM компонент, который устанавливается на компьютер, где работает внешнее ПО.

Требования к компьютеру для установки компонента BourneIdMgmt

- Компонент может быть установлен на компьютер с ОС Windows 7 и выше;
- Если программа BourneID, к которой будет выполнять запросы приложение, находится на другом компьютере локальной сети, то сетевые адаптеры обоих компьютеров должны поддерживать протокол TCP/IPv4.

2. Дистрибутив установки BourneID Management Pack

Запустите на выполнение дистрибутив **BourneIdMgmtPackSetup.exe**, который вы можете загрузить по ссылке: https://gg4u.ru/download/BourneIdMgmtPackSetup.exe.

Для успешной установки, учетная запись пользователя, от имени которого запускается программа установки, должна обладать правами локального администратора. В противном случае, имя и пароль пользователя с правами локального администратора будет запрошен программой установки. Установка выполнятся в несколько шагов.

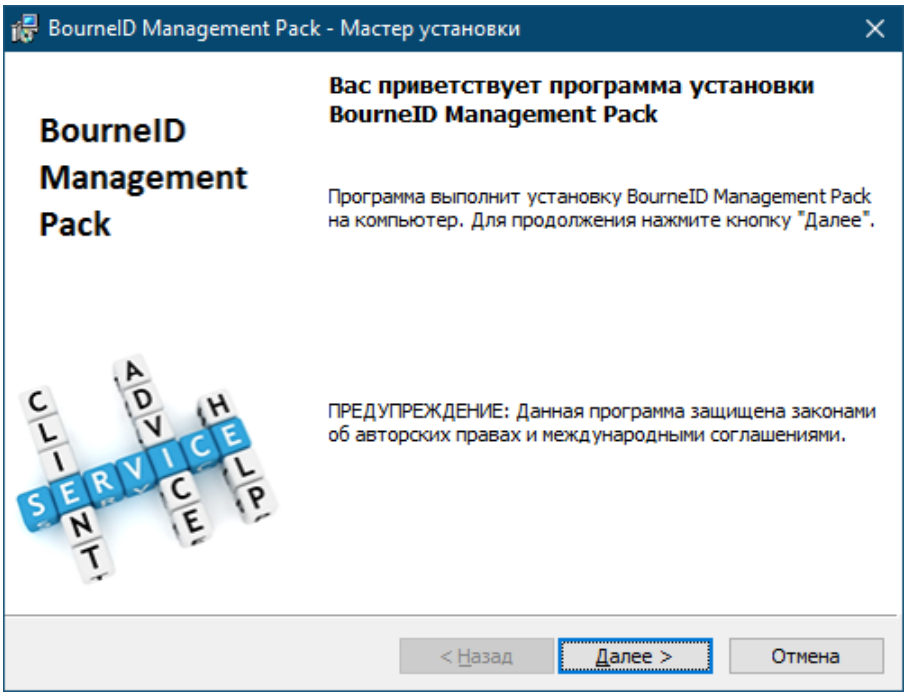

Рис. 1. Страница приветствия программы установки.

Нажмите кнопку "**Далее >**".

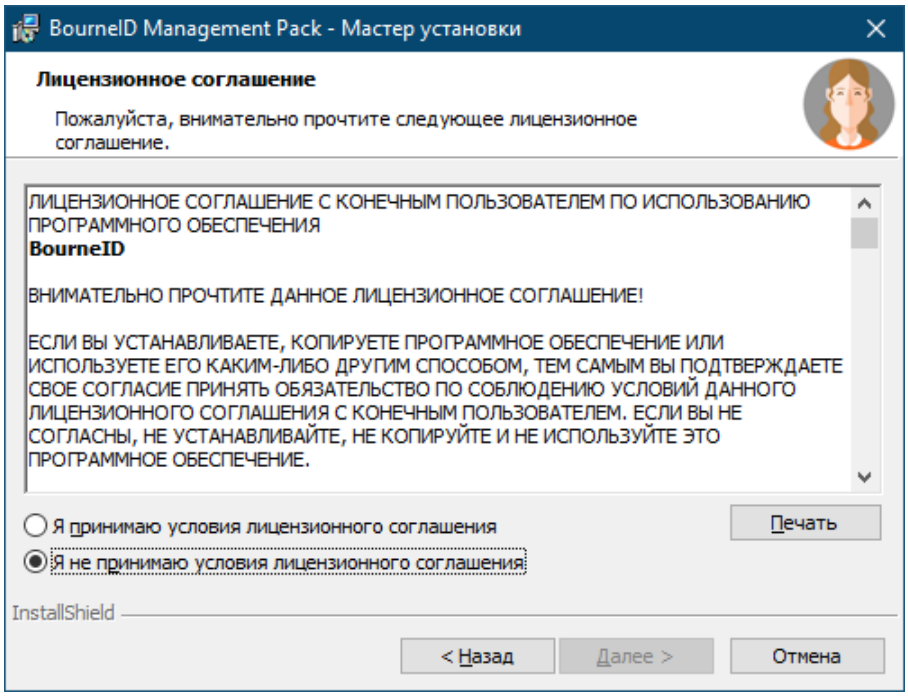

Рис. 2. Лицензионное соглашение.

Ознакомьтесь с содержанием лицензионного соглашения. Если Вы согласны с условиями соглашения, то выберите опцию "**Я принимаю условия**..." и нажмите кнопку "**Далее >**". В противном случае прекратите установку, нажав кнопку **Отмена**.

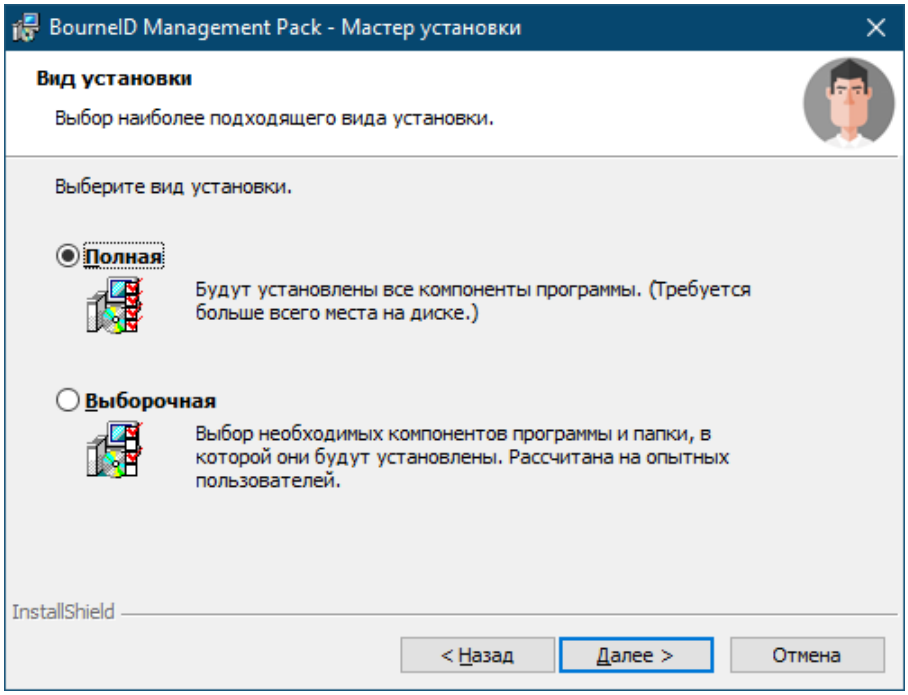

Рис. 3. Вид установки.

Дистрибутив включает 2 компонента:

1. **BourneID Management компонент**. Собственно, сам COM компонент. Обязателен для установки;

2. **Разработка**. Устанавливает руководство разработчика, демонстрационную программу ManagementClient32.exe и включаемые файлы для компиляции в среде Visual Studio. Установка **не обязательна** для компьютера конечного пользователя внешнего ПО.

Выберите тип установки "**Полная**" для установки всех компонентов в папки по умолчанию. Выбрав вид установки "**Выборочная**", вы можете отказаться от установки компонента "Разработка", а также изменить папки установки компонентов.

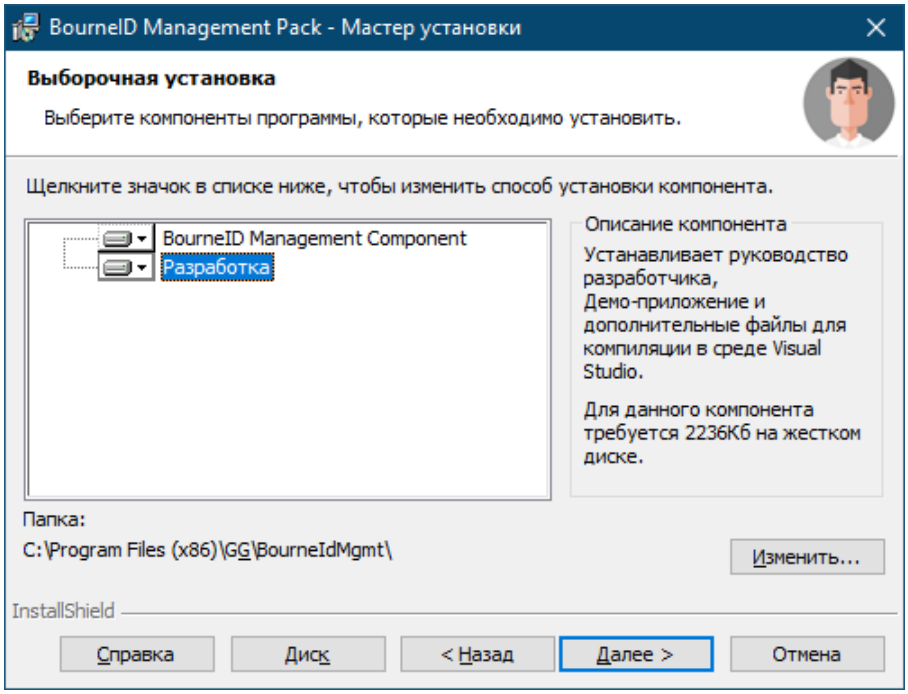

Рис. 4. Выборочная установка.

Нажмите кнопку "**Далее >**".

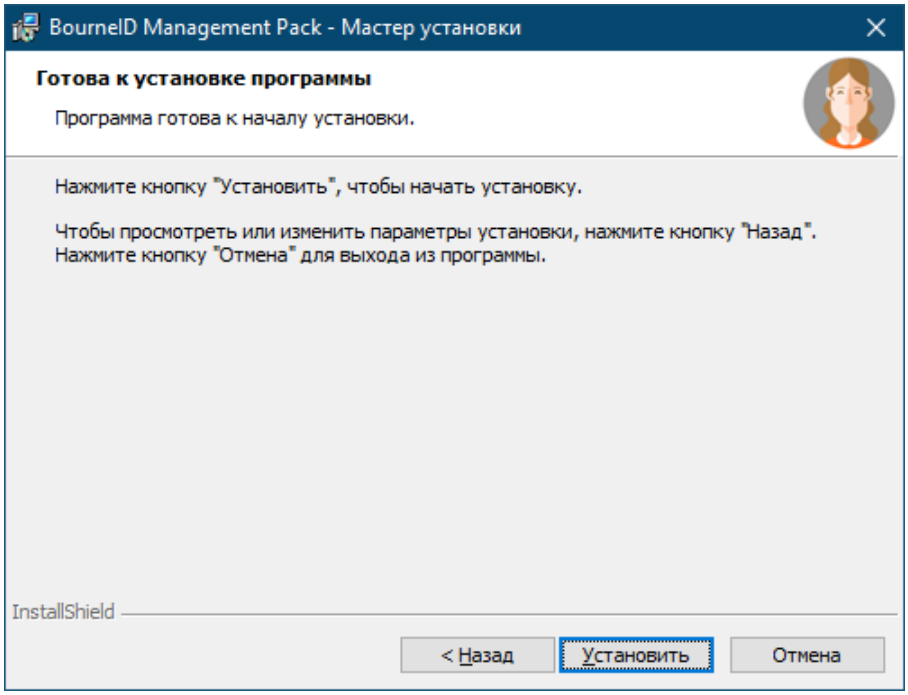

Рис. 5. Готовность к установке.

Нажмите кнопку **"Установить"** для начала установки. В процессе установки выводится информация о действиях, которые выполняются программой установки в данный момент. В случае ошибок (проблем в процессе установки) выводятся соответствующие сообщения, и пользователь может предпринять корректирующие действия.

В случае успешного завершения установки будет показано окно:

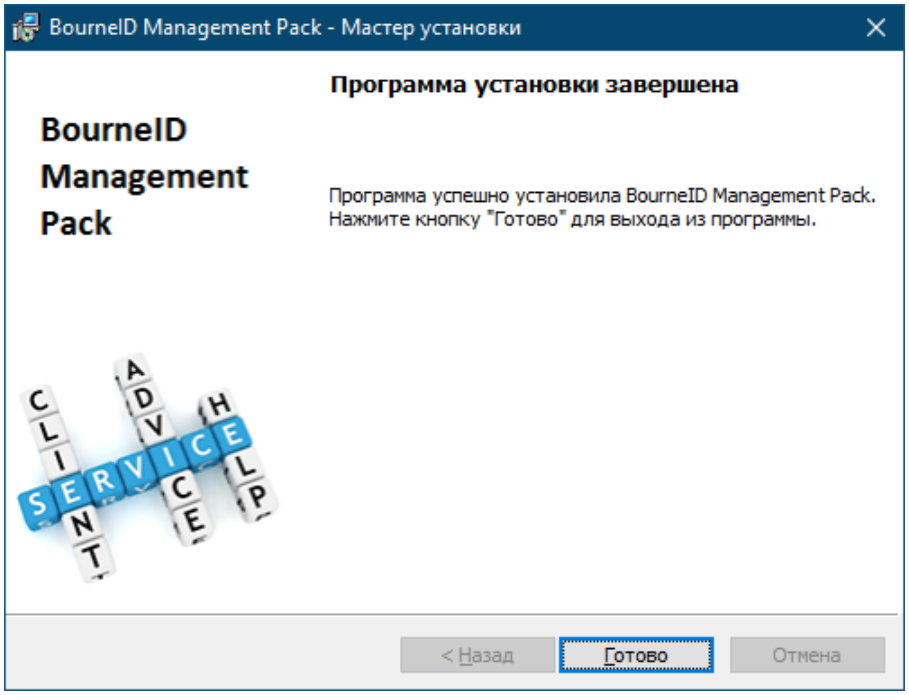

Рис. 6. BourneID Management Pack установлен.

3. Настройка программы BourneID

Запустите в работу программу BourneID. Работа с программой BourneID описана в отдельном документе **"BourneID - Руководстве пользователя"**. Подключитесь к камере. Убедитесь, что в настройках программы BourneID на закладке "Управление" включен Management Server и разрешена UDP локация. Если на комьютере работает Брандмауэр Windows, то разрешите в нем работу программы BourneID для локальной сети.

4. Запуск демонстрационной программы ManagementClient32.exe

Программа будет установлена на ваш компьютер, если установка была полной (устанавливался компонент "Разработка"). В зависимости от версии операционной системы, запуск производится через меню Windows:
Пуск → Все программы → G&G → BourneID → ManagemenPack → ManagementClient32 (Windows 7)
Пуск → Все программы → G&G → ManagementClient32 (Windows 8)
Пуск → G&G → ManagementClient32 (Windows 10)

Главное окно программы показано на рисунке 7.

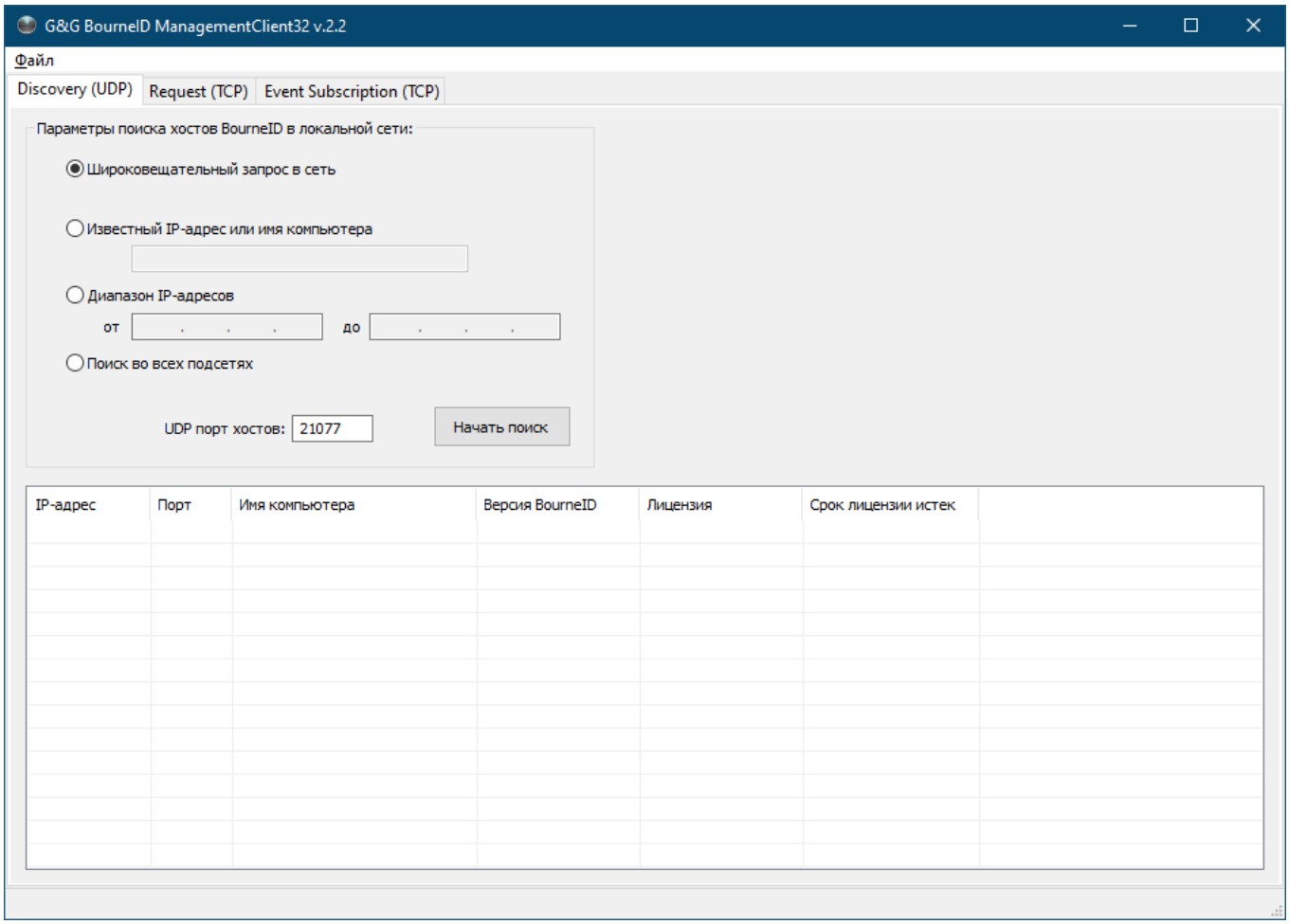

Рис. 7. Главное окно программы BourneID ManagementClient32.

Если на компьютере работает Брандмауэр Windows, то разрешите в нем работу программы ManagementClient32.exe для локальной сети.

В верхней части окно имеет 3 закладки: **"Discovery (UDP)", "Request (TCP)"** и **"Event Subscription (TCP)"**.

Поиск хостов BourneID

На закладке "Discovery (UDP)" вы можете выполнить поиск (локацию) компьютеров с выполняющимися программами BourneID.

Выберите вариант поиска:

- **Широковещательный запрос в сеть**;
- **Известный IP-адрес или имя компьютера**. Требует задания адреса или имени компьютера с программой BourneID;
- **Диапазон IP-адресов**. Требует задания начального и конечного IP-адресов диапазона поиска;
- **Поиск во всех подсетях**. Поиск в диапазонах масок IP-адресов всех сетевых интерфейсов компьютера.

Задайте номер UDP порта хостов (по умолчанию 21077) и нажмите кнопку **"Начать поиск"**.

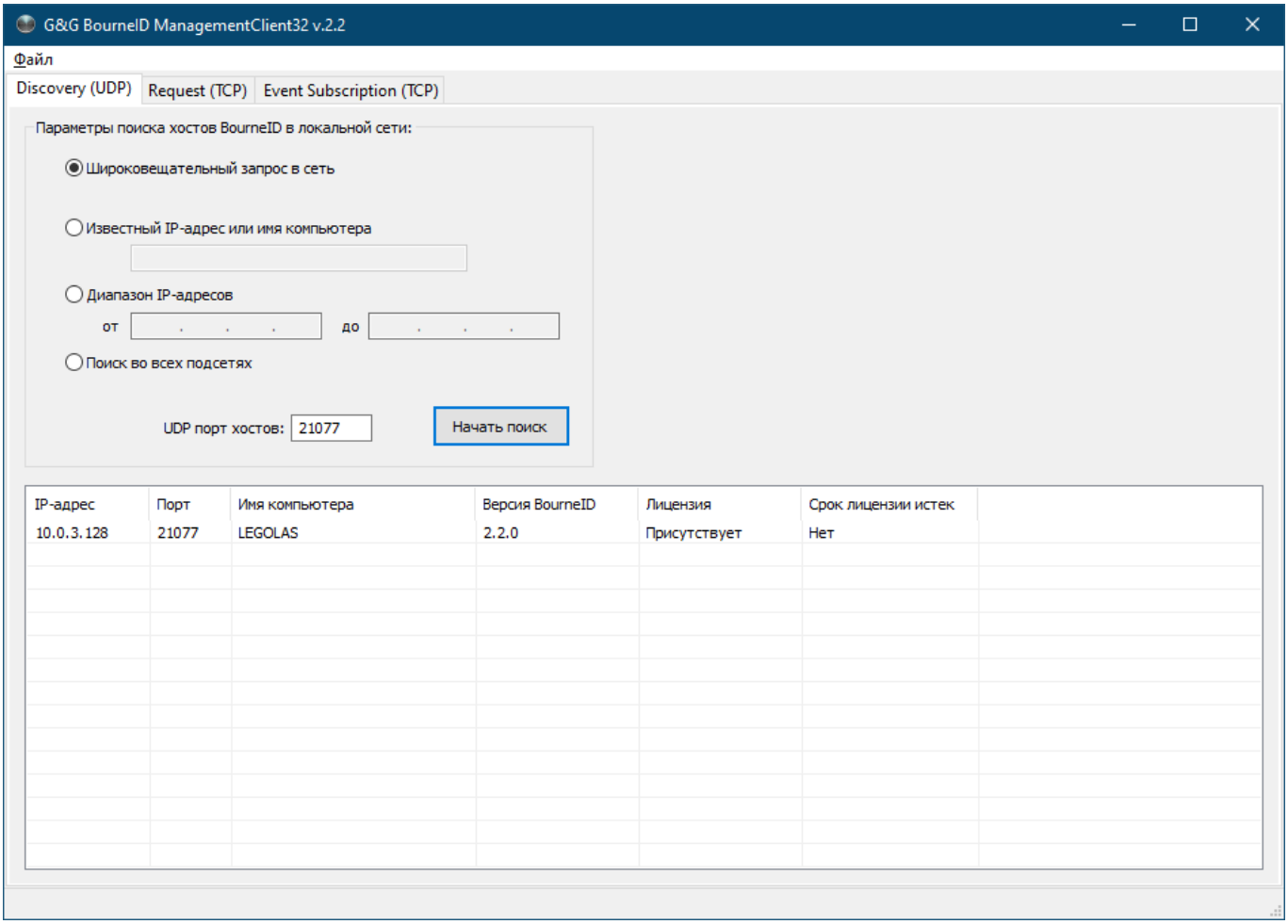

Рис. 8. Найден компьютер с работающей программой BourneID.

Для найденных хостов в списке отображаются: IP-адрес компьютера, номер порта, имя компьютера, версия программы BourneID, признак наличия лицензии, признак истечения срока лицензии.

Выполнение запросов к хостам BourneID

Откройте закладку **"Request (TCP)"**.

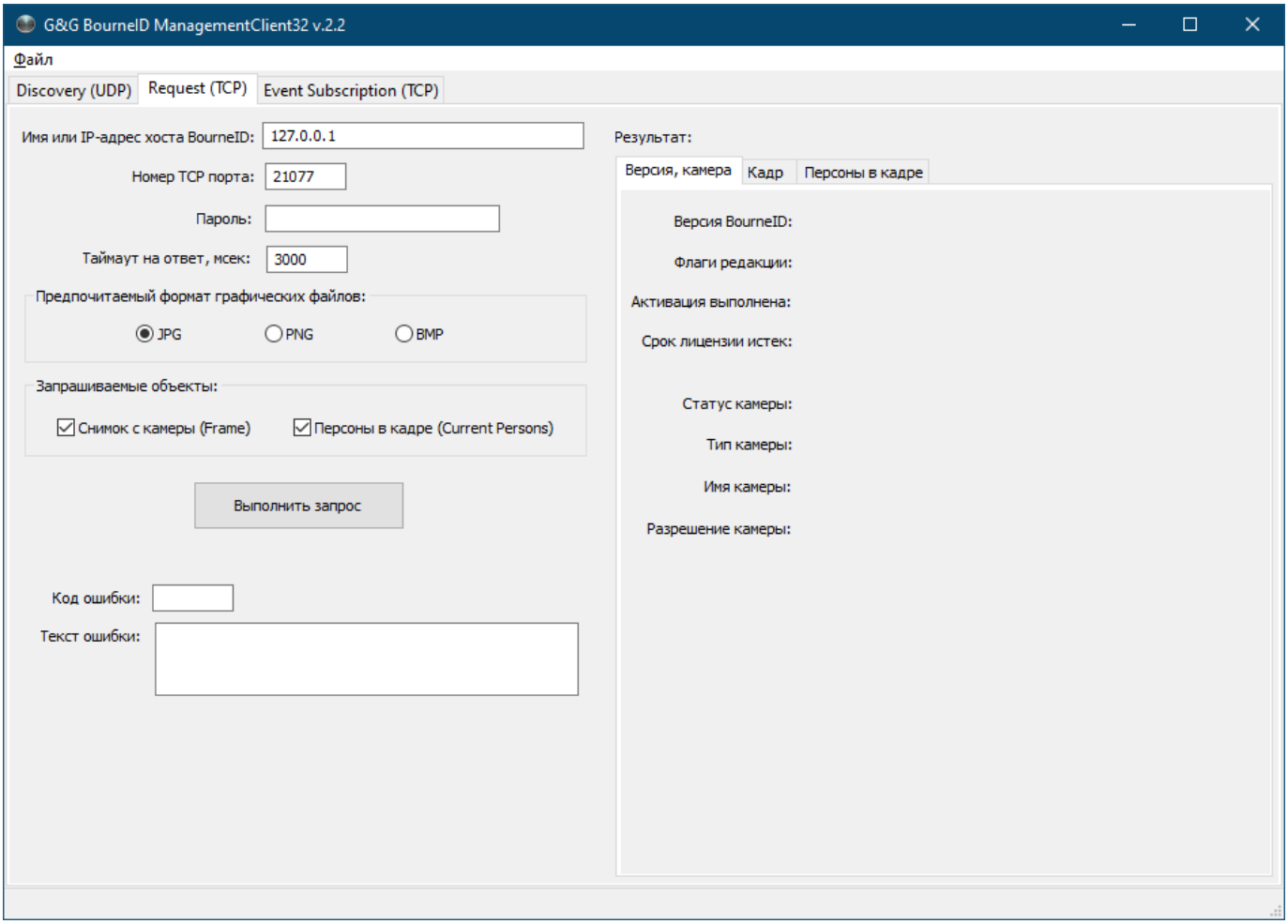

Рис. 9. Закладка "Request (TCP)".

1. Введите **имя или IP-адрес хоста BourneID**. По умолчанию запрос выполняется к **localhost** (127.0.0.1);
2. Уточните **номер TCP порта**, по которому Management Server ожидает запросы. По умолчанию 21077;
3. Введите **пароль** для Management Server, если он был задан в настройках программы BourneID;
4. **Таймаут на ответ, мсек**. Задает значение таймаута, в миллисекундах, на выполнение запроса к хосту BourneID и получение ответа. Значение по умолчанию 3000 мсек (3 секунды);
5. **Предпочитаемый формат графических файлов**. В ответе хоста BourneID на запрос присутствуют графические изображения. Эта опция задает их формат. Значение по умолчанию **JPG**. Другие возможные значения формата: **PNG** и **BMP**;
6. **Запрашиваемые объекты**. Определяет, какая информация запрашивается у хоста BourneID. Вы можете запрашивать: Текущий **снимок с камеры** (Frame), набор объектов – распознанных **персон в кадре** (Current Persons), либо и то и другое.

Нажмите кнопку **"Выполнить запрос"**.

В результате выполнения запроса, значение поля **"Код ошибки"** будет заполнено значением **0**, если запрос был удачным и хост BourneID вернул запрашиваемую информацию. Значение кода ошибки отличное от нуля свидетельствует о той или иной ошибке, возникшей при выполнении запроса. При этом поле **"Текст ошибки"** будет заполнено описанием причины ошибки.

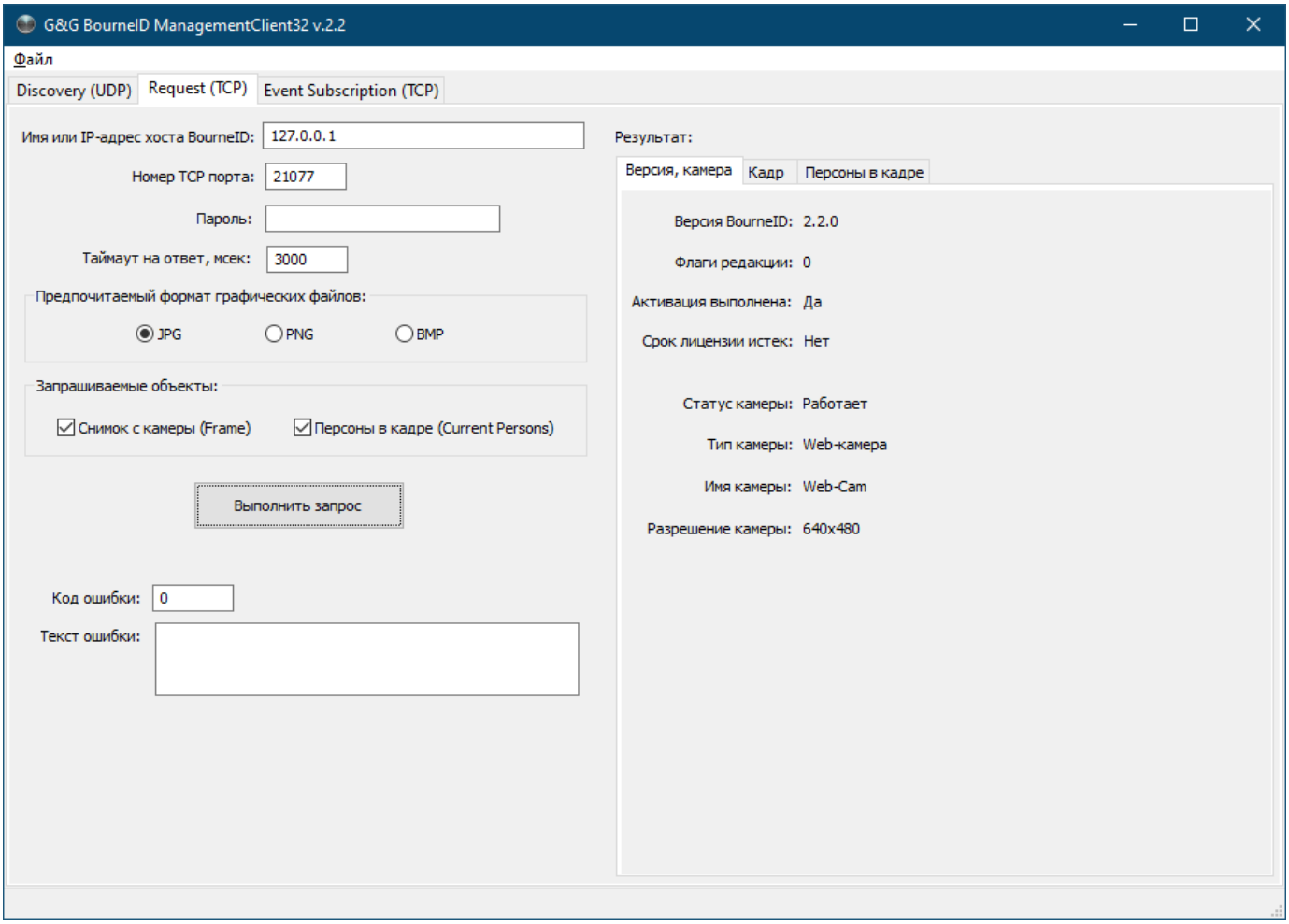

Рис. 10. Запрос к хосту BourneID успешно выполнен.

В случае успешного завершения запроса, хост BourneID возвращает:

- **Номер версии** программы BourneID, например, 2.2.0;
- **Флаг редакции.** Имеет значение 0. Зарезервировано для будущего использования;
- **Активация выполнена.** Имеет значение "Истина", если активация программы BourneID была выполнена, и "Ложь", в противном случае;
- **Срок лицензии истек.** Имеет значение "Истина", если активация не была выполнена, и срок пробного использования истек, либо если активация была выполнена с ограничением по дате (временная лицензия), и срок временной лицензии истек. Значение "Ложь" во всех других случаях;
- **Статус камеры.** Может принимать значения "Работает" или "Не работает";
- **Тип камеры.** Тип текущей работающей камеры. Может принимать значения "Web-камера" или "IP-камера ". Если камера не работает, то значение не имеет смысла;
- **Имя камеры**. Символьное имя (название) текущей работающей камеры. Если камера не работает, то значение не имеет смысла;
- **Разрешение камеры**. Размер кадра текущей работающей камеры в пикселях. Если камера не работает, то значение не имеет смысла.

Если в запрашиваемых объектах был выбран флаг запроса снимка с камеры (Frame), и есть текущая работающая камера, то открыв закладку "Кадр", в правой части окна программы, вы можете увидеть снимок с текущей камеры в момент запроса.

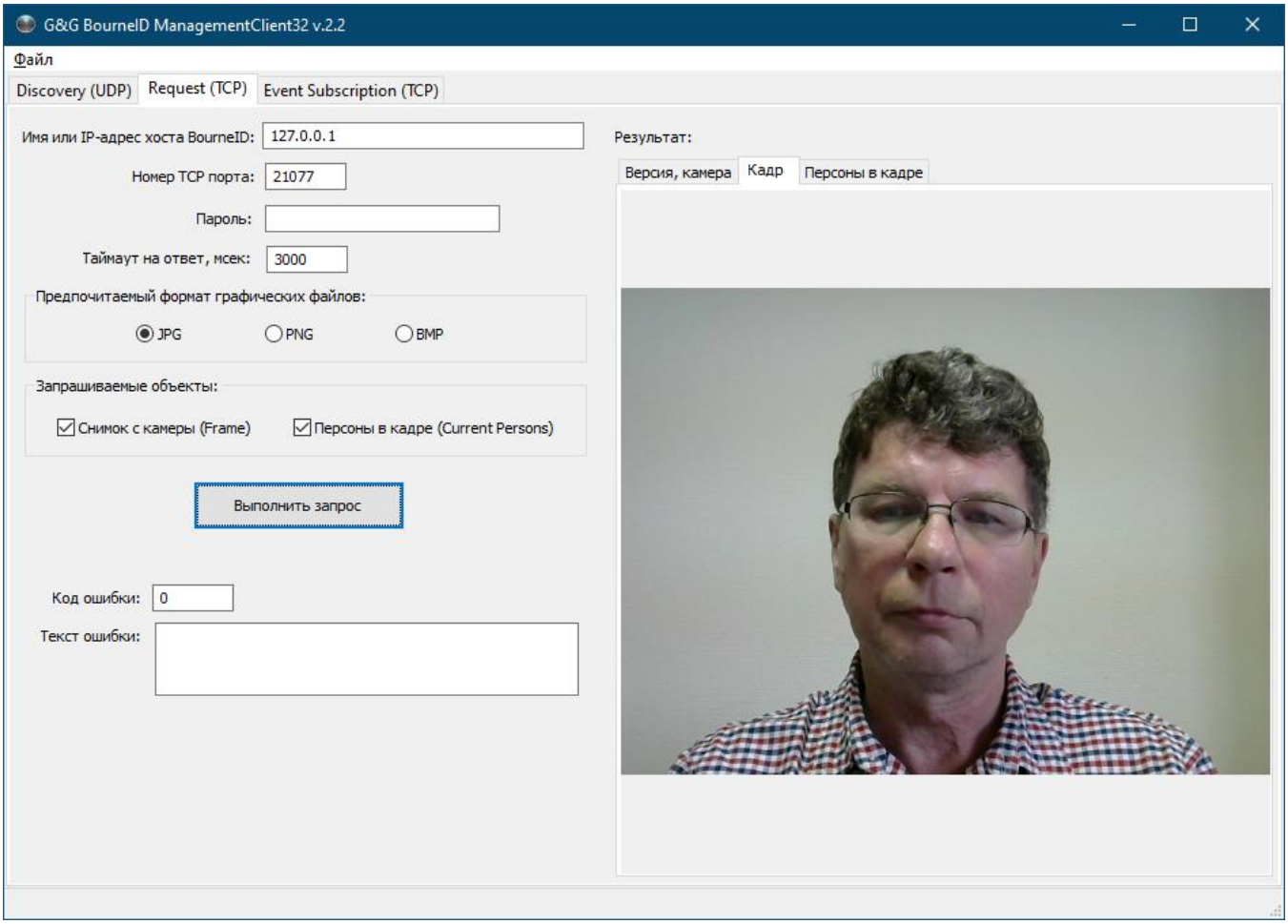

Рис. 11. Кадр камеры в момент запроса.

Если запрашивались объекты "Персоны в кадре (Current Persons)", то хост BourneID возвращает набор объектов "Персона в кадре".

"Персона в кадре":

- Объект был распознан программой BourneID, с вероятностью распознавание не ниже порога уверенного распознавания;
- Объект еще присутствует в кадре или объект отсутствует в кадре, но менее секунды.

Для просмотра набора персон в кадре откройте закладку "Персоны в кадре" (рис. 12).

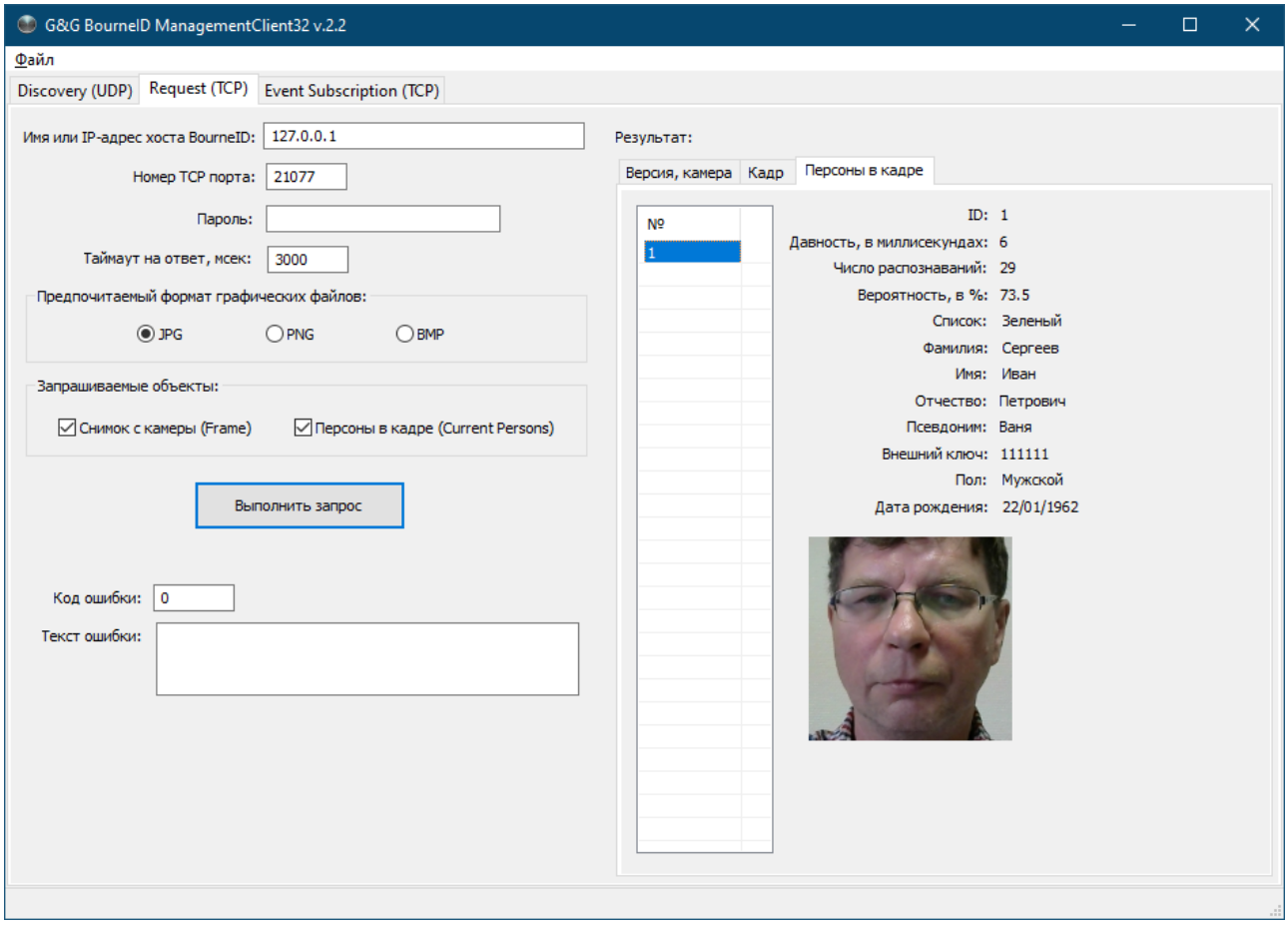

Рис. 12. Персоны в кадре.

Выберите в списке слева номер (индекс) персоны в наборе. Если список пуст, то персоны в кадре отсутствуют. Каждый объект "Персона в кадре" имеет свойства:

- **ID персоны**;
- **Давность, в миллисекундах**. Разность между текущим временем и временем, когда персона последний раз присутствовала в кадре;
- **Число распознаваний**. Сколько раз персона была распознана за все время присутствия;
- **Вероятность распознавания, в %**. Максимальное значение вероятности распознавания за время присутствия в кадре;
- **Список персоны**: Зеленый, Красный или Серый;
- **Фамилия**, **Имя**, **Отчество**, **Псевдоним**, **Внешний ключ**;
- **Пол** – мужской или женский;
- **Дата рождения**.

В интерфейсе программы BourneID запросы отображаются в панели сообщений на закладке "Нотификации".

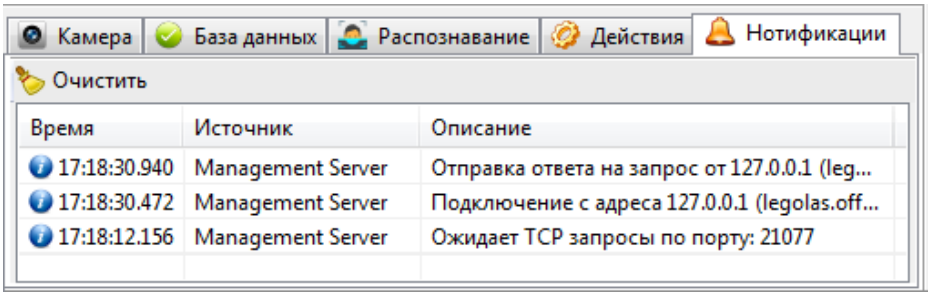

Рис. 13. Сообщения в запросах в интерфейсе программы BourneID.

Подписка на события BourneID

Откройте закладку **"Event Subscription (TCP)"**.

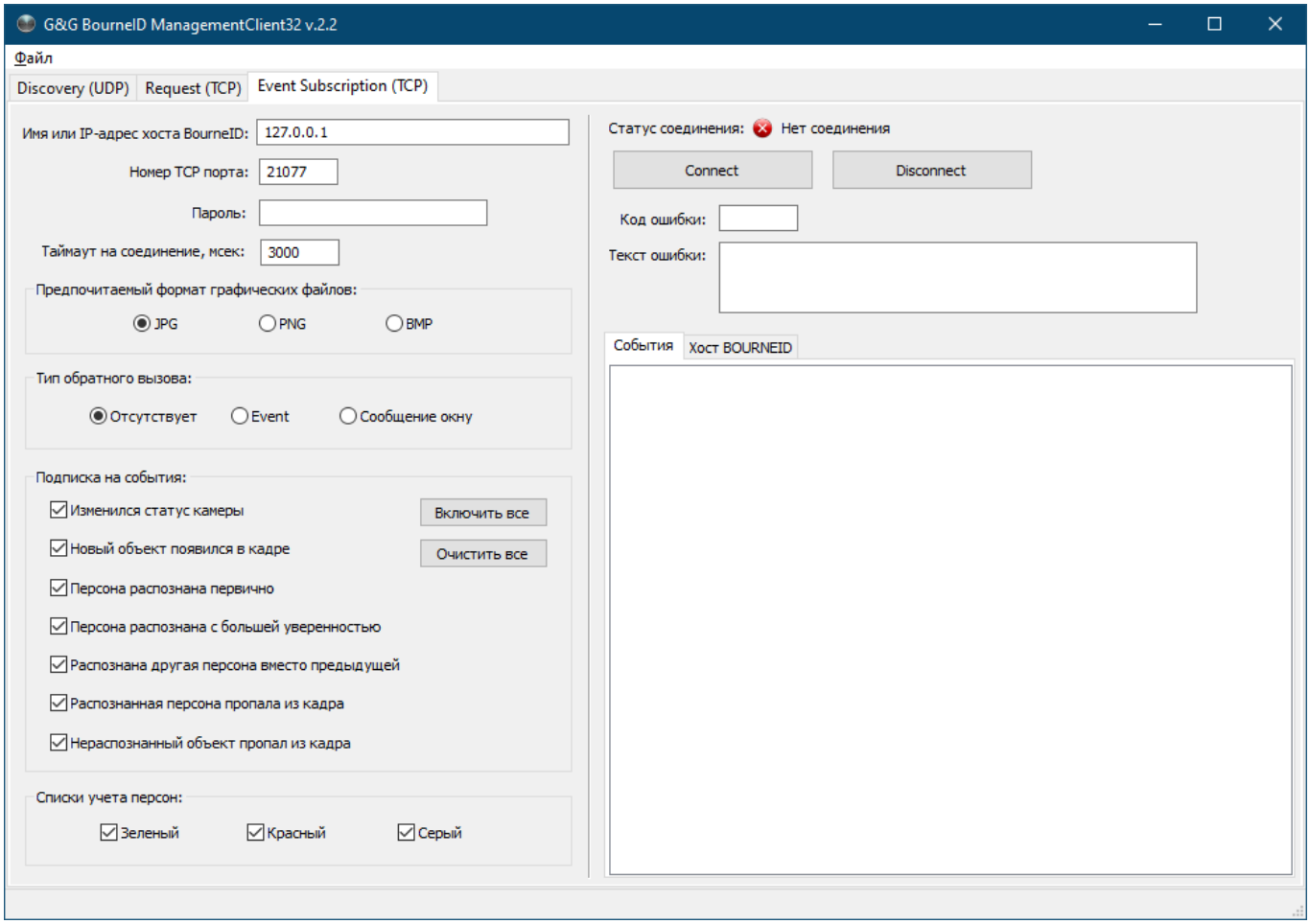

Рис. 14. Закладка "Event Subscription (TCP)".

- Введите **имя или IP-адрес хоста BourneID**. По умолчанию запрос выполняется к **localhost** (127.0.0.1);
- Уточните **номер TCP порта**, по которому Management Server ожидает запросы. По умолчанию 21077;
- Введите **пароль** для Management Server, если он был задан в настройках программы BourneID;
- **Таймаут на ответ, мсек**. Задает значение таймаута, в миллисекундах, на соединение с хостом BourneID и получение подтверждения подписки на события. Значение по умолчанию 3000 мсек;
- **Предпочитаемый формат графических файлов**. Некоторые события BourneID содержат графические изображения. Эта опция задает их формат. Значение по умолчанию **JPG**. Другие возможные значения формата: **PNG** и **BMP**;
- Выберите тип обратного вызова программы при наступлении событий. По умолчанию **Отсутствует**. Другие возможные значения: **Event** и **Сообщение окну**.
- **Подписка на события**. Определяет, какие типы событий BourneID будут передаваться программе:
  1. **Изменился статус камеры**. Камера может находиться в состояниях **IDDLE** (работа с камерой остановлена) или **WORKS** (камера работает). Для состояния WORKS дополнительно передается текущая операция камеры: Не определена, Стартована, Открыта, Закрыта, «Висит» или «Вновь доступна». События генерируются как при изменении состояния, так и операции;
  2. **Новый объект появился в камере**. Событие генерируется при определении лица человека в кадре. Каждый новый объект в камере получает номер (KeyNumber). С событием передается картинка области лица объекта;
  3. **Персона распознана первично**. Для нового объекта в камере произошло распознавание персоны с вероятностью выше порога уверенного распознавания. В параметрах события

передается KeyNumber, информация о персоне, список учета, значение вероятности распознавания и картинка области лица;

4. **Персона распознана с большей уверенностью.** Для уже распознанной персоны вероятность распознавания выше предыдущей. В параметрах события передается KeyNumber, информация о персоне, список учета, значение вероятности распознавания и картинка области лица;
5. **Распознана другая персона вместо предыдущей**. Вместо ранее распознанной персоны распознана другая. В параметрах события передается KeyNumber, идентификаторы предыдущей и текущей персоны и значения вероятностей распознавания;
6. **Распознанная персона пропала из кадра**. В параметрах события передается KeyNumber, идентификатор и список учета персоны, а также длительность присутствия персоны в кадре;
7. **Нераспознанный объект пропал из кадра**. В параметрах события передается KeyNumber и длительность присутствия объекта в кадре;

- **Списки учета персон: Зеленый, Красный, Серый**. Определяет, какие учетные списки интересуют в событиях, обладающих свойством списка.

Нажмите кнопку **"Connect"** для подключения к хосту BourneID и подписки на события с заданным набором параметров. В результате выполнения запроса, значение поля **"Код ошибки"** будет заполнено значением **0**, если соединение было удачным, и хост BourneID подтвердил подписку на события. Значение кода ошибки отличное от нуля свидетельствует об ошибке. При этом поле **"Текст ошибки"** будет заполнено описанием причины ошибки. В случае успешного подключения, в список событий будут помещаться описания и картинки событий, происходящих на хосте BourneID, типы которых удовлетворяют набору типов событий, запрошенных в подписке.

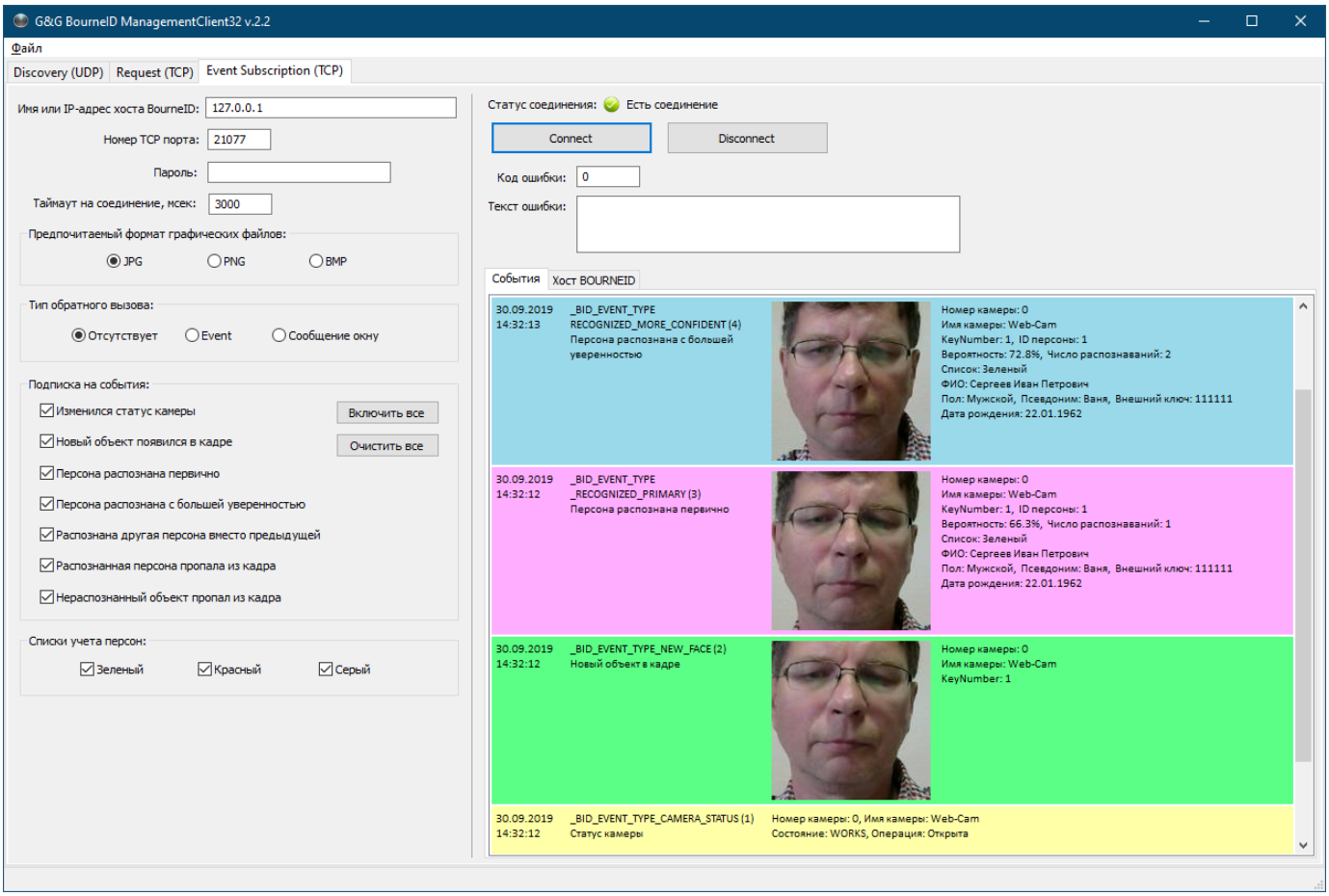

Рис. 15. Отображение событий по подписке.

Для отключения от хоста BourneID (и завершения подписки на события) нажмите кнопку **"Disconnect"**.

5. Объектная модель COM компонента

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

Компонент имеет два базовых класса:

- **Request**, позволяющий выполнять запросы к программе BourneID, получать и сохранять возвращаемую информацию о текущем состоянии камеры и распознанных в данный момент персонах в кадре;
- **EventSubscription**, позволяющий подписываться на события BourneID и получать их во внешнем ПО.

Объект Request

Имеет единственный интерфейс **IRequest**.

Создание объекта в коде приложения

VBScript

В скриптовых языках объект создается с использованием строкового эквивалента идентификатора интерфейса, имеющего значение `BourneIdMgmt.Request`

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

'Переменная будет содержать объект
Dim l_objRequest
'Создаем объект
Set l_objRequest = CreateObject("BourneIdMgmt.Request")

'Объект создан?
If Err.number <> 0 Then
    MsgBox "Ошибка создания объекта ' BourneIdMgmt.Request': " & _
              Err.Description, vbOKOnly + vbCritical, "Ошибка"
    Exit Sub
End If
...
'Удаляем объект
Set l_objRequest = Nothing
```

C++

Подключите в код программы файлы "BourneIdMgmt_i.c", "BourneIdMgmt_i.h" и "BourneIdMgmt_def.h", которые вы можете найти в подкаталоге CPP папки установки (по умолчанию "c:\Program Files (x86)\G&G\BourneIdMgmt\CPP"). Инициализируйте библиотеку OLE в коде потока.

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

#include "BourneIdMgmt_i.c"
#include "BourneIdMgmt_i.h"
#include "BourneIdMgmt_def.h"
```

```
...
CoInitialize(NULL);
...
// SMART указатель на объект с интерфейсом IRequest
CComPtr<IRequest> l_spIRequest;
HRESULT hr = l_spIRequest.CoCreateInstance(CLSID_Request);
if(hr == S_OK) {
    // Объект создан
}
else {
    // Ошибка создания объекта. Описание ошибки можно извлечь из hr
}
...
// Удаляем объект. Не обязательно, т.к. это smart указатель
l_spIRequest = NULL;
...
```

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

Свойства объекта Request

Объект имеет следующие свойства (Properties):

LastErrorCode

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

	Код ошибки	**Описание причины ошибки**
ERROR_SUCCESS	0	Нет ошибки
ERROR_INVALID_PARAMETER	87	Неверно задан параметр
ERROR_UNKNOWN_COMPONENT	1607	Ошибка создания объекта DOMDocument
ERROR_INVALID_FUNCTION	1	Ошибка создания XML содержимого запроса
ERROR_NOT_ENOUGH_MEMORY	8	Ошибка выделения памяти в системе
	10000	BourneID не ответил на запрос в заданное время
	102	Некорректный ответ BourneID на запрос
	101	Ошибка при выполнении внутренней операции.
	103	Ошибка сохранения графического файла
	От 10004 до 11004	Ошибки WinSocket
	Другие коды	Описание ошибки можно получить из свойства **LastErrorText**

VBScript

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

C++

```
HRESULT CRequest::get_LastErrorCode([out, retval] ULONG* pErrorCode);
```

```
...
LONG l_lLastErrorCode;
l_spIRequest->get_LastErrorCode(&l_lLastErrorCode);
// В l_lLastErrorCode теперь находится код ошибки последней операции
```

LastErrorText

READ only. Строка. Возвращает текст ошибки последней операции.

VBScript

```
Dim l_sLastErrorText
'Получаем описание ошибки последней операции
l_sLastErrorText = l_objRequest.LastErrorText
```

C++

```
HRESULT CRequest::get_LastErrorText([out, retval] BSTR* pErrorText);
```

```
...
CComBSTR l_bstrErrorText;
l_spIRequest->get_LastErrorText(&l_bstrErrorText);
// В l_bstrErrorText теперь находится описание ошибки
...
```

ServerHost

READ/WRITE. Строка. Получает или задает имя хоста BourneID. При создании объекта значение свойства устанавливается в "127.0.0.1". В качестве имени хоста может быть использована строка, содержащая IP-адрес, имя или доменное имя компьютера. Максимальная длина строки - 255 символов. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Неверно задано имя или IP-адрес хоста".

VBScript

```
Dim l_sOldServerHost
'Получаем текущее имя хоста BourneID
l_sOldServerHost = l_objRequest.ServerHost
'Устанавливаем другое имя хоста сервера
objRequest.ServerHost = "MYHOST.OFFICE.FABRICAM.COM"
```

C++

```
HRESULT CRequest::get_ServerHost([out, retval] BSTR* pCurrentServerHost);

HRESULT CRequest::put_ServerHost([in] BSTR newServerHost);
```

```
CComBSTR l_bstrOldServerHost;
hr = l_spIRequest->get_ServerHost(&l_bstrOldServerHost);
// Задаем другое имя хоста, например по его IP-адресу
hr = l_spIRequest->put_ServerHost(CComBSTR("10.0.3.124"));
...
```

ServerPort

READ/WRITE. Целое число в диапазоне от 0 до 65535. Получает или задает номер порта TCP, по которому хост BourneID ожидает соединений. При создании объекта значение свойства устанавливается в 21077 (номер порта BourneID по умолчанию).

VBScript

```
Dim l_nOldServerPort
'Получаем текущее значение
nOldServerPort = l_objRequest.ServerPort
'Устанавливаем другой номер порта
objRequest.ServerPort = 8008
```

C++

```
HRESULT CRequest::get_ServerPort([out, retval] USHORT* pCurrentServerPort);

HRESULT CRequest::put_ServerPort([in] USHORT newServerPort);
```

```
...
USHORT l_suOldServerPort;
l_spIRequest->get_ServerPort(&l_suOldServerPort);
// Задаем другой номер порта TCP
l_spIRequest->put_ServerPort(8008);
...
```

ServerPassword

READ/WRITE. Строка. Получает или задает пароль для запроса к хосту BourneID. Пароль чувствителен к регистру. При создании объекта, значение пароля устанавливается как пустая строка. При задании пароля, максимальная длина строки составляет 15 символов. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Пароль слишком длинный".

VBScript

```
Dim l_sOldPassword
'Получаем текущий пароль
l_sOldPassword = l_objRequest.ServerPassword
```

```
'Устанавливаем другой пароль
objRequest.ServerPassword = "don't worry"
```

C++

```
HRESULT CRequest::get_ServerPassword([out, retval] BSTR* pCurrentServerPassword);

HRESULT CRequest::put_ServerPassword([in] BSTR newServerPassword);
```

```
...
CComBSTR l_bstrOldPassword;
hr = l_spIRequest->get_ServerPassword(&l_bstrOldPassword);
// Задаем другой пароль
hr = l_spIRequest->put_ServerPassword(CComBSTR("don't worry"));
...
```

PreferredImgFormat

READ/WRITE. Целое число. Получает или задает формат графических изображений, которые будет возвращать хост BourneID в ответе на запрос. Формат может иметь значения: 0 – JPG, 1 – PNG или 2 – BMP. При создании объекта значение свойства устанавливается в 0 (формат JPG). Если при задании свойства будет задано число отличное от 0, 1 или 2, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Неверно задан формат графических файлов".

Константы определены в BourneIdMgmt_def.h

```
#define _IMG_FILE_JPG_FORMAT            0
#define _IMG_FILE_PNG_FORMAT            1
#define _IMG_FILE_BMP_FORMAT            2
```

VBScript

```
Const IMG_FILE_JPG_FORMAT = 0
Const IMG_FILE_PNG_FORMAT = 1
Const IMG_FILE_BMP_FORMAT = 2

Dim l_nOldImgFormat
'Получаем текущее значение
l_nOldImgFormat = l_objRequest.PreferredImgFormat
'Задаем другой формат
objRequest.PreferredImgFormat = IMG_FILE_PNG_FORMAT
```

C++

```
HRESULT CRequest::get_PreferredImgFormat([out, retval] ULONG* pCurrentFormat);

HRESULT CRequest::put_PreferredImgFormat([in] ULONG newFormat);
```

```
...
ULONG l_ulOldImgFormat;
l_spIRequest->get_PreferredImgFormat(&l_ulOldImgFormat);
// Задаем другой другой формат
l_spIRequest->put_PreferredImgFormat(_IMG_FILE_PNG_FORMAT);
```

ReplyTimeout

READ/WRITE. Целое число. Получает или задает значение таймаута, в миллисекундах, на ожидание ответа на запрос от хоста BourneID. При создании объекта значение свойства устанавливается в 3000 (3 секунды). Минимальное значение таймаута – 1000 (1 секунда), максимальное – 10000 (10 секунд). Если при задании свойства, будет задано значение меньше чем 1000, то значение таймаута будет установлено в 1000. При задании значения большего, чем 10000, значение таймаута будет установлено в 10000.

VBScript

```
Dim l_nOldTimeout
'Получаем текущее значение
l_nOldTimeout = l_objRequest.ReplyTimeout
'Задаем другое значение
objRequest.ReplyTimeout = 5000
```

C++

```
HRESULT CRequest::get_ReplyTimeout([out, retval] ULONG* pCurrentTimeout);

HRESULT CRequest::put_ReplyTimeout([in] ULONG newTimeout);
```

```
...
ULONG l_ulOldTimeout;
l_spIRequest->get_ReplyTimeout(&l_ulOldTimeout);
// Задаем другое значение
l_spIRequest->put_ReplyTimeout(5000);
...
```

Примечание! Последующие свойства объекта относятся к информации, возвращаемой хостом BourneID в ответе на запрос. Свойства имеют атрибут READ ONLY и могут быть запрошены после вызова метода **Request**, закончившегося успехом.

Version

READ only. Строка. Возвращает номер версии программы BourneID, содержащийся в ответе на запрос к хосту BourneID. Формат строки: *<MajorNumber>.<MinorNumber>.<BuildNumber>*, например, "2.2.0".

VBScript

```
Dim l_sBourneIDVersion
'Получаем версию BourneID
l_sBourneIDVersion = l_objRequest.Version
```

C++

```
HRESULT CRequest::get_Version([out, retval] BSTR* pVersionText);
```

```
...
CComBSTR l_bstrBourneIDVersion;
l_spIRequest->get_Version(&l_bstrBourneIDVersion);
...
```

EditionFlag

READ only. Целое число. Зарезервировано для последующего использования. В текущей версии возвращает значение 0.

Registered

READ only. Логическая величина. Возвращает TRUE, если активация программы BourneID была выполнена, и FALSE, в противном случае.

VBScript

```
Dim l_boolBourneIDRegistered
l_boolBourneIDRegistered = l_objRequest.Registered
```

C++

```
HRESULT CRequest::get_Registered([out, retval] VARIANT_BOOL* pIsRegistered);
```

```
...
VARIANT_BOOL l_vbBourneIDRegistered;
l_spIRequest->get_Registered(&l_vbBourneIDRegistered);
...
```

LicExpired

READ only. Логическая величина. Возвращает TRUE, если активация программа BourneID не выполнена и срок пробного использования истек, либо если активация была выполнена с ограничением по дате (временная лицензия), и срок временной лицензии истек. Возвращает FALSE во всех других случаях.

VBScript

```
Dim l_boolLicExpired
l_boolLicExpired = l_objRequest.LicExpired
```

C++

```
HRESULT CRequest::get_LicExpired([out, retval] VARIANT_BOOL* pIsLicExpired);
```

```
...
VARIANT_BOOL l_vbLicExpired;
l_spIRequest->get_LicExpired(&l_vbLicExpired);
...
```

CameraWorks

READ only. Логическая величина. Возвращает TRUE, если камера в программе BourneID в момент запроса работает. Возвращает FALSE, если камера остановлена.

VBScript

```
Dim l_boolCameraWorks
l_boolCameraWorks = l_objRequest.CameraWorks
```

C++

```
HRESULT CRequest::get_CameraWorks([out, retval] VARIANT_BOOL* pIsCameraWorks);
```

```
...
VARIANT_BOOL l_vbCameraWorks;
l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

if (l_vbCameraWorks) {
    // Можно запрашивать другую информацию по результату запроса
    ...
}
...
```

CameraType

READ only. Целое число. Возвращает 0, если текущая работающая камера имеет тип Web-камера, и 1, если это IP-камера. Значение валидно, только если свойство CameraWorks имеет значение TRUE.

VBScript

```
Dim l_boolCameraWorks
l_boolCameraWorks = l_objRequest.CameraWorks

If l_boolCameraWorks = True Then
    Dim l_nCameraType
    'Получаем тип камеры
    l_nCameraType = l_objRequest.CameraType
    ...
End If
...
```

C++

```
HRESULT CRequest::get_CameraType([out, retval] ULONG* pType);
```

```
#define _WEB_CAMERA_TYPE          0
#define _IP_CAMERA_TYPE           1
...
VARIANT_BOOL l_vbCameraWorks;
l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

if (l_vbCameraWorks) {
```

```
    // Получаем тип камеры
    ULONG l_ulCameraType;
    l_spIRequest->get_CameraType(&l_ulCameraType);
    if (l_ulCameraType == _WEB_CAMERA_TYPE) {
        // Web-камера
        ...
    }
    else {
        // IP-камера
        ...
    }
    ...
}
...
```

CameraName

READ only. Строка. Возвращает имя текущей камеры. Значение валидно, только если свойство CameraWorks имеет значение TRUE.

VBScript

```
Dim l_boolCameraWorks
l_boolCameraWorks = l_objRequest.CameraWorks

If l_boolCameraWorks = True Then
    Dim l_sCameraName
    'Получаем имя камеры
    l_sCameraName = l_objRequest.CameraName
    ...
End If
...
```

C++

```
HRESULT CRequest::get_CameraName([out, retval] BSTR* pName);
```

```
...
VARIANT_BOOL l_vbCameraWorks;
l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

if (l_vbCameraWorks) {
    // Получаем имя камеры
    CComBSTR l_bstrCameraName;
    l_spIRequest->get_CameraName(&l_bstrCameraName);
    ...
}
...
```

CameraHorzResolution

READ only. Целое число. Возвращает горизонтальное разрешение текущей камеры, в пикселях. Значение валидно, только если свойство CameraWorks имеет значение TRUE.

VBScript

```
...
Dim l_nCameraHorzRes
l_nCameraHorzRes = l_objRequest.CameraHorzResolution
...
```

C++

```
HRESULT CRequest::get_CameraHorzResolution([out, retval] ULONG* pResolution);
```

```
...
ULONG l_ulCameraHorzRes;
l_spIRequest->get_CameraHorzResolution(&l_ulCameraHorzRes);
...
```

CameraVertResolution

READ only. Целое число. Возвращает вертикальное разрешение текущей камеры, в пикселях. Значение валидно, только если свойство CameraWorks имеет значение TRUE.

VBScript

```
...
Dim l_nCameraVertRes
l_nCameraVertRes = l_objRequest.CameraVertResolution
...
```

C++

```
HRESULT CRequest::get_CameraVertResolution([out, retval] ULONG* pResolution)
```

```
...
ULONG l_ulCameraVertRes;
l_spIRequest->get_CameraVertResolution(&l_ulCameraVertRes);
...
```

CameraImageFile

READ only. Строка. Возвращает полный путь к графическому файлу, в котором сохранен текущий кадр камеры в момент запроса. Значение валидно, только если:

- При выполнении запроса к хосту BourneID, в параметре метода Request был задан флаг запроса текущего снимка камеры (Frame);
- Запрос завершился успехом;
- Свойство CameraWorks имеет значение TRUE.

Формат файла (JPG, PNG либо BMP) определяется значением свойства `PreferredImgFormat` заданным до момента выполнения запроса. Файл находится во временной папке учетной записи пользователя. Код программы не должен удалять файл с диска. Файл будет автоматически удален при удалении объекта или выполнении нового запроса к хосту BourneID.

VBScript

```
Dim l_boolCameraWorks
l_boolCameraWorks = l_objRequest.CameraWorks

If l_boolCameraWorks = True Then
    Dim l_sCameraFrameFile
    'Получаем имя файла с текущим кадром камеры
    l_ sCameraFrameFile = l_objRequest.CameraImageFile
    ...
End If
...
```

C++

```
HRESULT CRequest::get_CameraImageFile([out, retval] BSTR* pFileName);
```

```
...
VARIANT_BOOL l_vbCameraWorks;
l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

if (l_vbCameraWorks) {
    // Получаем имя файла с текущим кадром камеры
    CComBSTR l_bstrCameraFileName;
    l_spIRequest->get_CameraImageFile(&l_bstrCameraFileName);
    ...
}
...
```

CurrentPersonCount

READ only. Строка. Возвращает число распознанных персон (CurrentPerson) возвращенных в ответе на запрос. Значение валидно, только если:

- При выполнении запроса к хосту BourneID, в параметре метода Request был задан флаг запроса персон в кадре (Current Persons);
- Запрос завершился успехом;
- Свойство CameraWorks имеет значение TRUE.

В ответ на запрос хост BourneID возващает набор "Распознанных персон". Свойство `CurrentPersonCount` возвращает число элементов в наборе.

VBScript

```
Dim l_boolCameraWorks
l_boolCameraWorks = l_objRequest.CameraWorks

If l_boolCameraWorks = True Then
    Dim l_nPersonCount
    'Получаем число персон в наборе
    l_nPersonCount = l_objRequest.CurrentPersonCount
    ...
End If
...
```

C++

```
HRESULT CRequest::get_CurrentPersonCount([out, retval] LONG* pCount);
```

```
...
VARIANT_BOOL l_vbCameraWorks;
l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

if (l_vbCameraWorks) {
    // Получаем число персон в наборе
    LONG l_lPersonCount;
    l_spIRequest->get_CurrentPersonCount(&l_lPersonCount);
    ...
}
...
```

Методы объекта Request

Объект имеет следующие методы (Methods):

Request

```
HRESULT CRequest::Request([in] ULONG uServiceType);
```

Выполняет запрос к хосту BourneID и ожидает ответа с набором возвращаемых данных.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства LastErrorCode, а тест ошибки - LastErrorText.

uServiseType

Числовой параметр, определяющий, какую информацию будет возвращать хост BourneID. Значение параметра может иметь значения:
_REQUEST_TYPE_FRAME – возвращать снимок текущего кадра камеры
_REQUEST_TYPE_CURRENT_PERSONS – возвращать набор объектов **CurrentPerson**
_REQUEST_TYPE_ALL – возвращать снимок и набор объектов

Константы определены в файле BourneIdMgmt_def.h

```
#define _REQUEST_TYPE_FRAME              0x00000001
#define _REQUEST_TYPE_CURRENT_PERSONS    0x00000002
#define _REQUEST_TYPE_ALL   (_REQUEST_TYPE_FRAME | _REQUEST_TYPE_CURRENT_PERSONS)
```

При выполнении запроса используются значения следующих свойств объекта, заданных явно, или со значениями по умолчанию:

- **ServerHost** – Имя или IP-адрес компьютера с программой BourneID. По умолчанию *localhost*;
- **ServerPort** – Номер порта TCP, по которому хост BourneID ожидает запросы. По умолчанию *21077*;
- **ServerPassword** – Пароль для запроса к хосту BourneID. По умолчанию *пустой пароль*;
- **PreferredImgFormat** – Сообщает хосту BourneID, в каком формате нужно возвращать графические изображения. По умолчанию *JPG;*
- **ReplyTimeout** – Время ожидания ответа от хоста BourneID. По умолчанию *3000 мсек.*

При выполнении метода, компонент:

- Открывает TCP соединение;
- Передает пакет запроса;
- Ожидает ответа от хоста BourneID;
- Заполняет значения свойств объекта Request, связанные с возвращаемой информацией;
- Сохраняет графические изображения во временные файлы;
- Создает массив (набор) элементов "Распознанных персон", если такая информация присутствует в ответе.

Если все стадии завершаются успешно, то метод возвращает значение **S_OK** (0). Приложение может запрашивать значения полученных свойств, а также объекты **CurrentPerson** из полученного набора.

Если при выполнении метода на любой стадии фиксируется ошибка, то дальнейшей выполнение прекращается, и возвращается значение **S_FALSE** (1). Код ошибки, а также текстовое описание причины ошибки можно извлечь из значений свойств **LastErrorCode** и **LastErrorText**.

Код ошибки	Описание причины ошибки
1	"Неправильный пароль" "Ошибка создания XML содержимого запроса"
2	"Неправильная структура XML запроса. Отсутствует тег <Type>."
3	"Неподдерживаемый тип запроса" "Неверно задан параметр запрашиваемого сервиса (ServiceType)" "Неверно задан формат графических файлов (PreferredImgFormat)"
8	"Ошибка выделения памяти в системе"
87	"Неверно задан параметр запрашиваемого сервиса" "Невозможно определить IP-адрес хоста"
101	"Неизвестная ошибка select() на ожидании входящих данных." "Неизвестная ошибка при разборе XML ответа" "Ошибка при разборе XML ответа: <........>"
102	"Непонятный "мусор" в ответе на запрос." "Размер данных ответа на запрос слишком велик." "Тело ответа не является XML документом допустимого формата" "XML ответа не содержит корневой элемент" "Корневой элемент XML ответа имеет неправильное имя" "Отсутствует тег <ErrorCode> в XML ответа" "Отсутствует тег <BourneID> в XML ответа" "Отсутствует тег <Camera> в XML ответа"
103	"Ошибка сохранения графического файла с кадром камеры"
10000	"BourneID не ответил на запрос в заданное время"
От 10004 до 11004	Ошибки WinSocket "Открытие сокета: ...." "Привязка сокета: ...." "Задание размера буфера приема сокета: ...." "Задание размера буфера передачи сокета: ...."

	"Перевод сокета в Blocking режим: ...." "Установка соединения с хостом BourneID: ...." "Посылка запроса на хост BourneID: ...." "Ожидание входящих данных в ответ на запрос: ...." "Чтение данных ответа на запрос: ...."
Другие коды	Описание ошибки можно получить из значения свойства **LastErrorText**

VBScript

```
Const REQUEST_TYPE_FRAME = 1
Const REQUEST_TYPE_CURRENT_PERSONS = 2
Const REQUEST_TYPE_ALL = 3

Dim l_objRequest
'Создаем объект
Set l_objRequest = CreateObject("BourneIdMgmt.Request")

'Задаем значения для некоторых свойств
l_objRequest.ServerHost = "MYHOST.OFFICE.FABRICAM.COM"
l_objRequest.ServerPassword = "don't worry"

'Выполняем запрос
l_objRequest.Request REQUEST_TYPE_ALL

'В скриптовых языках метод не имеет возвращаемого значения.
'Поэтому, чтобы понять, как завершился запрос, проверяем код последней ошибки.
If l_objRequest.LastErrorCode <> 0 Then
    MsgBox "Ошибка (" & l_objRequest.LastErrorCode & ") запроса к хосту BourneID': " _
    & l_objRequest.LastErrorText, vbOKOnly + vbCritical, "Ошибка"
Else
   'Запрашиваем значения возвращенных свойств
    Dim l_bCameraWorks : l_bCameraWorks = l_objRequest.CameraWorks
    If l_bCameraWorks Then
        Dim l_sCameraName : l_sCameraName = l_objRequest.CameraName
        ...
        ...
    End If
End If
...
```

C++

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

#include "BourneIdMgmt_i.c"
#include "BourneIdMgmt_i.h"
#include "BourneIdMgmt_def.h"

...
CoInitialize(NULL);
...
// SMART указатель на объект с интерфейсом IRequest
CComPtr<IRequest> l_spIRequest;
HRESULT hr = l_spIRequest.CoCreateInstance(CLSID_Request);

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

```
    l_spIRequest->put_ServerHost(CComBSTR("MYHOST.OFFICE.FABRICAM.COM"));
    l_spIRequest->put_ServerPassword(CComBSTR("don't worry"));

    // Выполняем запрос
    hr = l_spIRequest->Request(_REQUEST_TYPE_ALL);

    if (hr != S_OK) {
        CComBSTR l_bstrErrorText;
        l_spIRequest->get_LastErrorText(&l_bstrErrorText);
        MessageBox(GetFocus(), _bstr_t(l_bstrErrorText),
                   "Ошибка запроса к хосту BourneID", MB_OK | MB_ICONSTOP);
    }
    else {
        // Запрашиваем значения возвращенных свойств
        VARIANT_BOOL l_vbCameraWorks;
        l_spIRequest->get_CameraWorks(&l_vbCameraWorks);

        if (l_vbCameraWorks) {
            CComBSTR l_bstrCameraName;
            l_spIRequest->get_CameraName(&l_bstrCameraName);
            ...
        }
    }
}
...
```

GetCurrentPerson

```
HRESULT CRequest::GetCurrentPerson(
    [in] LONG lIndex,
    [out, retval] ICurrentPerson** ppCurrentPerson);
```

Создает объект **CurrentPerson**, по заданному индексу массива "Распознанных персон", возвращенных по запросу.

- 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 персон".

lIndex

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

ppCurrentPerson

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

Объекты CurrentPerson в коде внешнего ПО **не создаются напрямую**. Вместо этого, у объекта Request запрашивается элемент массива "Распознанных персон". Полученный от метода объект (интерфейс объекта) **должен быть удален в коде приложения**.

VBScript

```
'Выполняем запрос
l_objRequest.Request REQUEST_TYPE_ALL

If l_objRequest.LastErrorCode = 0 Then
   'Запрашиваем число элементов массива ”Распознанных персон”
    Dim l_nPersonCount : l_nPersonCount = l_objRequest.CurrentPersonCount
    If l_nPersonCount > 0 Then
        Dim l_objCurrentPerson
        'Запрашиваем объект с индексом 0
        Set l_objCurrentPerson = l_objRequest.GetCurrentPerson 0
        'Запрашиваем свойства объекта CurrentPerson
        Dim l_nPersonID : l_nPersonID = l_objCurrentPerson.PersonID
        ...
        'Удаляем объект
        Set l_objCurrentPerson = Nothing
    End If
End If
...
```

C++

```
...
// Выполняем запрос к BourneID
hr = l_spIRequest->Request(_REQUEST_TYPE_ALL);

if (hr == S_OK) {
    // Запрашиваем число элементов массива ”Распознанных персон”
    LONG l_lPersonCount = 0;
    l_spIRequest->get_CurrentPersonCount(&l_lPersonCount);

    if (l_lPersonCount > 0) {
        CComPtr <ICurrentPerson> l_spCurrentPerson = NULL;

        // Запрашиваем интерфейс объекта с индексом 0
        if (S_OK == l_spIRequest->GetCurrentPerson(0, &l_spCurrentPerson))
        {
            // Запрашиваем свойства объекта CurrentPerson
            LONG l_lPersonID;
            l_spCurrentPerson->get_PersonID(&l_lPersonID);
            ...
        }
    }
}
...
```

Объект CurrentPerson

Объект, хранящий информацию о "Распознанной персоне" при выполнении запроса. Объект имеет только свойства, и не имеет методов. Объект имеет единственный интерфейс **ICurrentPerson**. Все свойства объекта имеют атрибуты READ ONLY.

Использование объектов в коде приложения

Объекты CurrentPerson создаются при вызове метода **GetCurrentPerson** объекта Request. Метод возвращает интерфейс на созданный объект. По окончании использования объекта, его необходимо удалить. Для автоматического удаления объекта CurrentPerson можно использовать smart указатель на интерфейс или явно вызывать Release() наследованного интерфейса IUnknown.

Свойства объекта CurrentPerson

PersonID

Целое число. Идентификатор персоны в базе данных программы BourneID.

VBScript

```
'Выполняем запрос к BourneID
l_objRequest.Request 3   'REQUEST_TYPE_ALL

If l_objRequest.LastErrorCode = 0 Then
   'Запрашиваем число элементов массива "Распознанных персон"
    Dim l_nPersonCount : l_nPersonCount = l_objRequest.CurrentPersonCount
    If l_nPersonCount > 0 Then
        Dim l_objCurrentPerson
        'Запрашиваем объект CurrentPerson с индексом 0
        Set l_objCurrentPerson = l_objRequest.GetCurrentPerson 0
        'Получаем идентификатор персоны
        Dim l_nPersonID : l_nPersonID = l_objCurrentPerson.PersonID
        ...
        'Удаляем объект
        Set l_objCurrentPerson = Nothing
    End If
End If
...
```

C++

```
HRESULT CCurrentPerson::get_PersonID([out, retval] LONG* pPersonID);
```

```
...
// Выполняем запрос к BourneID
hr = l_spIRequest->Request(_REQUEST_TYPE_ALL);

if (hr == S_OK) {
    // Запрашиваем число элементов массива "Распознанных персон"
    LONG l_lPersonCount = 0;
    l_spIRequest->get_CurrentPersonCount(&l_lPersonCount);

    if (l_lPersonCount > 0) {
        // Используем smart указатель на интерфейс
        CComPtr<ICurrentPerson> l_spCurrentPerson;

        // Запрашиваем интерфейс объекта CurrentPerson с индексом 0
```

```
        if (S_OK == l_spIRequest->GetCurrentPerson(0, &l_spCurrentPerson))
        {
            // Получаем идентификатор персоны
            LONG l_lPersonID;
            l_pCurrentPerson->get_PersonID(&l_lPersonID);

            ...
            // Удаляем объект CurrentPerson
            l_spCurrentPerson = NULL;
        }
        ...
    }
}
...
```

PresenceAge

Целое число. Давность присутствия персоны в кадре, в миллисекундах от момента последнего процессинга кадра.

VBScript

```
'Запрашиваем объект CurrentPerson с индексом 0
Set l_objCurrentPerson = l_objRequest.GetCurrentPerson 0
'Получаем давность присутствия персоны в кадре
Dim l_nPersonPresenceAge
l_nPersonPresenceAge = l_objCurrentPerson.PresenceAge
...
```

C++

```
HRESULT CCurrentPerson::get_PresenceAge([out, retval] LONG* pAgeMsec);
```

```
...
CComPtr<ICurrentPerson> l_spCurrentPerson;

// Запрашиваем интерфейс объекта CurrentPerson с индексом 0
if (S_OK == l_spIRequest->GetCurrentPerson(0, &l_spCurrentPerson))
{
    // Получаем давность присутствия персоны в кадре
    LONG l_lPersonPresenceAge;
    l_pCurrentPerson->get_PersonID(&l_lPersonPresenceAge);

    ...
    // Удаляем объект CurrentPerson
    l_spCurrentPerson = NULL;
}
...
```

RecognitionCount

Целое число. Число распознаваний. Сколько раз персона была распознана за все время присутствия в кадре.

VBScript

```
'Запрашиваем объект CurrentPerson с индексом 0
Set l_objCurrentPerson = l_objRequest.GetCurrentPerson 0
```

```
'Получаем число разпознаваний
Dim l_nPersonRecognitionCount
l_nPersonRecognitionCount = l_objCurrentPerson.RecognitionCount
...
```

C++

```
HRESULT CCurrentPerson::get_RecognitionCount([out, retval] LONG* pCount);
```

```
...
CComPtr<ICurrentPerson> l_spCurrentPerson;

// Запрашиваем интерфейс объекта CurrentPerson с индексом 0
if (S_OK == l_spIRequest->GetCurrentPerson(0, &l_spCurrentPerson))
{
    // Получаем число разпознаваний
    LONG l_lPersonRecognitionCount;
    l_pCurrentPerson->get_RecognitionCount(&l_lPersonRecognitionCount);

    ...
    // Удаляем объект CurrentPerson
    l_spCurrentPerson = NULL;
}
...
```

ConfidencePercent

Вещественное число двойно точности (double). Вероятность (уверенность) распознавания, в %. Максимальное значение вероятности распознавания за время присутствия.

VBScript

```
'Запрашиваем объект CurrentPerson с индексом 0
Set l_objCurrentPerson = l_objRequest.GetCurrentPerson 0
'Получаем вероятность разпознавания
Dim l_nPersonConfidencePercent
l_nPersonConfidencePercent = l_objCurrentPerson.ConfidencePercent
...
```

C++

```
HRESULT CCurrentPerson::get_ConfidencePercent([out, retval] DOUBLE* pPercent);
```

```
...
CComPtr<ICurrentPerson> l_spCurrentPerson;

// Запрашиваем интерфейс объекта CurrentPerson с индексом 0
if (S_OK == l_spIRequest->GetCurrentPerson(0, &l_spCurrentPerson))
{
    // Получаем вероятность разпознавания
    double l_dblPersonConfidencePercent;
    l_pCurrentPerson->get_ConfidencePercent(&l_dblPersonConfidencePercent);

    ...
    // Удаляем объект CurrentPerson
    l_spCurrentPerson = NULL;
}
...
```

ListNum

Целое число. **Список персоны**: Зеленый, Красный или Серый.

Константы определены в файле BourneIdMgmt_def.h

```
#define _PERSON_GREEN_LIST        0
#define _PERSON_RED_LIST          1
#define _PERSON_GRAY_LIST         2
```

VBScript

```
Const PERSON_GREEN_LIST = 0
Const PERSON_RED_LIST = 1
Const PERSON_GRAY_LIST = 2

'Получаем список персоны
Dim l_nPersonListNum
l_nPersonListNum = l_objCurrentPerson.ListNum
...
```

C++

```
HRESULT CCurrentPerson::get_ListNum([out, retval] LONG* pListNum);
```

```
...
LONG l_lPersonListNum;
l_pCurrentPerson->get_ConfidencePercent(&l_lPersonListNum);

switch (lPersonListNum) {
    case _PERSON_GREEN_LIST:
        break;
    case _PERSON_RED_LIST:
        break;
    case _PERSON_GRAY_LIST:
        break;
}
...
```

Surname

Строка. Фамилия.

VBScript

```
'Получаем фамилию персоны
Dim l_sPersonSurname
l_sPersonSurname = l_objCurrentPerson.Surname
...
```

C++

```
HRESULT CCurrentPerson::get_Surname([out, retval] BSTR* pSurname);
```

```
...
```

```
CComBSTR l_bstrPersonSurname;
l_pCurrentPerson->get_Surname(&l_bstrPersonSurname);
...
```

Forename

Строка. Имя.

VBScript

```
'Получаем имя персоны
Dim l_sPersonForename
l_sPersonForename = l_objCurrentPerson.Forename
...
```

C++

```
HRESULT CCurrentPerson::get_Forename([out, retval] BSTR* pForename);
```

```
...
CComBSTR l_bstrPersonForename;
l_pCurrentPerson->get_Forename(&l_bstrPersonForename);
...
```

Secondname

Строка. Отчество.

VBScript

```
Dim l_sPersonSecondname
l_sPersonSecondname = l_objCurrentPerson.Secondname
...
```

C++

```
HRESULT CCurrentPerson::get_Secondname([out, retval] BSTR* pSecondname);
```

```
...
CComBSTR l_bstrPersonSecondname;
l_pCurrentPerson->get_Secondname(&l_bstrPersonSecondname);
...
```

Alias

Строка. Псевдоним персоны.

VBScript

```
Dim l_sPersonAlias
l_sPersonAlias = l_objCurrentPerson.Alias
...
```

C++

```
HRESULT CCurrentPerson::get_Alias([out, retval] BSTR* pAlias);
```

```
...
CComBSTR l_bstrPersonAlias;
l_pCurrentPerson->get_Alias(&l_bstrPersonAlias);
...
```

ExtKey

Строка. Внешний ключ персоны.

VBScript

```
Dim l_sPersonExtKey
l_sPersonExtKey = l_objCurrentPerson.ExtKey
...
```

C++

```
HRESULT CCurrentPerson::get_ExtKey([out, retval] BSTR* pExtKey);
```

```
...
CComBSTR l_bstrPersonExtKey;
l_pCurrentPerson->get_ExtKey(&l_bstrPersonExtKey);
...
```

Gender

Целое число. **Пол персоны**: мужской или женский.

Константы определены в файле BourneIdMgmt_def.h

```
#define _GENDER_FEMALE      0
#define _GENDER_MALE        1
```

VBScript

```
Const GENDER_FEMALE = 0
Const GENDER_MALE = 1

'Получаем пол персоны
Dim l_nPersonGender
l_nPersonGender = l_objCurrentPerson.Gender
...
```

C++

```
HRESULT CCurrentPerson::get_Gender([out, retval] LONG* pGender);
```

```
...
LONG l_lPersonGender;
l_pCurrentPerson->get_Gender(&l_lPersonGender);
...
```

DateOfBirth

Дата. **Дата рождения**. Если для персоны дата рождения не задана, то значение равно 0 (нулевая дата).

VBScript

```
'Получаем дату рождения
Dim l_dtPersonDOB
l_dtPersonDOB = CDate(l_objCurrentPerson.DateOfBirth)
If CDbl(l_dtPersonDOB) > 0.0  Then
    'Дата рождения присутствует
End If
...
```

C++

```
HRESULT CCurrentPerson::get_DateOfBirth([out, retval] DATE* pDOB);
```

```
...
DATE l_dtPersonDOB;
l_pCurrentPerson->get_DateOfBirth(&l_dtPersonDOB);
if (l_dtPersonDOB > 0.0) {
    // Дата рождения присутствует. Конвертируем в SYSTEMTIME
    UDATE uDate;
    VarUdateFromDate((DATE) l_dtPersonDOB, 0, &uDate);
    ...
}
...
```

ImageFile

Строка. Возвращает полный путь к графическому файлу, в котором сохранено лицо распознанной персоны в момент, когда вероятность распознавания имела максимальное значение. Размер изображения 150x150 пикселей. Формат файла (JPG, PNG либо BMP) определяется значением свойства `PreferredImgFormat` объекта Request, заданным до момента выполнения запроса. Файл находится во временной папке учетной записи пользователя. Код программы не должен удалять файл с диска. Файл будет автоматически удален при удалении объекта.

VBScript

```
Dim l_sPersonFaceImgFile
‘ Получаем имя файла с лицом персоны
l_sPersonFaceImgFile = l_objCurrentPerson.ImageFile
...
```

C++

```
HRESULT CCurrentPerson::get_ImageFile([out, retval] BSTR* pImageFile);
```

```
...
// Получаем имя файла с лицом персоны
CComBSTR l_bstrPersonFaceImgFile;
l_pCurrentPerson->get_ImageFile(&l_bstrPersonFaceImgFile);
...
```

Объект EventSubscription

Используется для подписки на события. Имеет единственный интерфейс **IEventSubscription**.

Создание объекта в коде приложения

VBScript

В скриптовых языках объект создается с использованием строкового эквивалента идентификатора интерфейса, имеющего значение `BourneIdMgmt.EventSubscription`

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

'Переменная будет содержать объект
Dim l_objEventSubscription
'Создаем объект
Set l_objEventSubscription = CreateObject("BourneIdMgmt.EventSubscription")

'Объект создан?
If Err.number <> 0 Then
    MsgBox "Ошибка создания объекта 'BourneIdMgmt.EventSubscription': " & _
              Err.Description, vbOKOnly + vbCritical, "Ошибка"
    Exit Sub
End If
...
'Удаляем объект
Set l_objEventSubscription = Nothing
```

C++

Подключите в код программы файлы "BourneIdMgmt_i.c", "BourneIdMgmt_i.h" и "BourneIdMgmt_def.h", которые вы можете найти в подкаталоге CPP папки установки (по умолчанию "c:\Program Files (x86)\G&G\BourneIdMgmt\CPP"). Инициализируйте библиотеку OLE в коде потока.

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

#include "BourneIdMgmt_i.c"
#include "BourneIdMgmt_i.h"
#include "BourneIdMgmt_def.h"

...
CoInitialize(NULL);
...
// SMART указатель на объект с интерфейсом IEventSubscription
CComPtr<IEventSubscription> l_spEventSubscription;
HRESULT hr = l_spEventSubscription.CoCreateInstance(CLSID_EventSubscription);
if(hr == S_OK) {
    // Объект создан
}
else {
    // Ошибка создания объекта. Описание ошибки можно извлечь из hr
}
...
// Удаляем объект
l_spEventSubscription = NULL;
...
```

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

Свойства объекта EventSubscription

Объект имеет следующие свойства (Properties):

LastErrorCode

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

	Код ошибки	Описание причины ошибки
ERROR_SUCCESS	0	Нет ошибки
ERROR_INVALID_PARAMETER	87	Неверно задан параметр
ERROR_UNKNOWN_COMPONENT	1607	Ошибка создания объекта DOMDocument
ERROR_INVALID_FUNCTION	1	Ошибка создания XML содержимого запроса
ERROR_NOT_ENOUGH_MEMORY	8	Ошибка выделения памяти в системе
	10000	BourneID не ответил на запрос в заданное время
	102	Некорректный пакет данных от BourneID
	101	Ошибка при выполнении внутренней операции.
	103	Ошибка сохранения графического файла
	От 10004 до 11004	Ошибки WinSocket
ERROR_CONNECTION_ACTIVE	1230	Соединение уже установлено
ERROR_HOST_UNREACHABLE	1232	Невозможно определить IP-адрес хоста BourneID
ERROR_CONTENT_BLOCKED	1296	Блокировано при соединении
ERROR_XML_PARSE_ERROR	1465	Ошибка при разборе XML описания события
ERROR_NOT_CONNECTED	2250	Соединение не установлено
ERROR_OBJECT_NOT_FOUND	4312	Ошибка создания объекта
ERROR_NO_MORE_ITEMS	259	Нет событий в списке
	Другие коды	Описание ошибки можно получить из свойства **LastErrorText**

VBScript

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

C++

```
HRESULT CEventSubscription::get_LastErrorCode([out, retval] ULONG* pErrorCode);
```

```
...
LONG l_lLastErrorCode;
l_spEventSubscription->get_LastErrorCode(&l_lLastErrorCode);
// В l_lLastErrorCode теперь находится код ошибки последней операции
```

LastErrorText

READ only. Строка. Возвращает текст ошибки последней операции.

VBScript

```
Dim l_sLastErrorText
'Получаем описание ошибки последней операции
l_sLastErrorText = l_objEventSubscription.LastErrorText
```

C++

```
HRESULT CEventSubscription::get_LastErrorText([out, retval] BSTR* pErrorText);
```

```
...
CComBSTR l_bstrErrorText;
l_spEventSubscription->get_LastErrorText(&l_bstrErrorText);
// В l_bstrErrorText теперь находится описание ошибки
...
```

ServerHost

READ/WRITE. Строка. Получает или задает имя хоста BourneID. При создании объекта значение свойства устанавливается в "127.0.0.1". В качестве имени хоста может быть использована строка, содержащая IP-адрес, имя или доменное имя компьютера. Максимальная длина строки 255 символов. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Неверно задано имя или IP-адрес хоста". При попытке изменить значение после установки соединения код ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст последней ошибки в " Блокировано при соединении".

VBScript

```
Dim l_sOldServerHost
'Получаем текущее имя хоста BourneID
l_sOldServerHost = l_objEventSubscription.ServerHost
'Устанавливаем другое имя хоста сервера
l_objEventSubscription.ServerHost = "MYHOST.OFFICE.FABRICAM.COM"
```

C++

```
HRESULT CEventSubscription::get_ServerHost([out, retval] BSTR* pCurrentServerHost);

HRESULT CEventSubscription::put_ServerHost([in] BSTR newServerHost);
```

```
CComBSTR l_bstrOldServerHost;
l_spEventSubscription->get_ServerHost(&l_bstrOldServerHost);
// Задаем другое имя хоста, например по его IP-адресу
l_spEventSubscription->put_ServerHost(CComBSTR("10.0.3.124"));
...
```

ServerPort

READ/WRITE. Целое число в диапазоне от 0 до 65535. Получает или задает номер порта TCP, по которому хост BourneID ожидает соединений. При создании объекта значение свойства устанавливается в 21077 (номер порта BourneID по умолчанию). При попытке изменить значение порта после установки соединения код ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст последней ошибки в "Блокировано при соединении".

VBScript

```
Dim l_nOldServerPort
'Получаем текущее значение
nOldServerPort = l_objEventSubscription.ServerPort
'Устанавливаем другой номер порта
l_objEventSubscription.ServerPort = 8008
```

C++

```
HRESULT CEventSubscription::get_ServerPort([out, retval] USHORT* pCurrentServerPort);

HRESULT CEventSubscription::put_ServerPort([in] USHORT newServerPort);
```

```
...
USHORT l_suOldServerPort;
l_spEventSubscription->get_ServerPort(&l_suOldServerPort);
// Задаем другой номер порта TCP
l_spEventSubscription->put_ServerPort(8008);
...
```

ServerPassword

READ/WRITE. Строка. Получает или задает пароль для запроса к хосту BourneID. Пароль чувствителен к регистру. При создании объекта, значение пароля устанавливается как пустая строка. При задании пароля, максимальная длина строки составляет 15 символов. Если будет передана строка большей длины, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Пароль слишком длинный". При попытке изменить значение пароля после установки соединения код ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст последней ошибки в "Блокировано при соединении".

VBScript

```
Dim l_sOldPassword
'Получаем текущий пароль
l_sOldPassword = l_objEventSubscription.ServerPassword
'Устанавливаем другй пароль
l_objEventSubscription.ServerPassword = "don’t worry"
```

C++

```
HRESULT CEventSubscription::get_ServerPassword([out, retval] BSTR* pServerPassword);

HRESULT CEventSubscription::put_ServerPassword([in] BSTR newServerPassword);
```

```
...
CComBSTR l_bstrOldPassword;
l_spEventSubscription->get_ServerPassword(&l_bstrOldPassword);
// Задаем другой пароль
hr = l_spEventSubscription->put_ServerPassword(CComBSTR("don’t worry"));
...
```

PreferredImgFormat

READ/WRITE. Целое число. Получает или задает формат графических изображений, которые будет возвращать хост BourneID и объект EventSubscription. Формат может иметь значения: 0 – JPG, 1 – PNG или 2 – BMP.

При создании объекта значение свойства устанавливается в 0 (формат JPG). Если при задании свойства будет задано число отличное от 0, 1 или 2, то текущее значение свойства не изменится, а код ошибки последней операции будет установлен в ERROR_INVALID_PARAMETER, а текст последней ошибки в "Неверно задан формат графических файлов". При попытке изменить значение после установки соединения код ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст последней ошибки в "Блокировано при соединении".

Константы графических форматов определены в BourneIdMgmt_def.h

```
#define _IMG_FILE_JPG_FORMAT            0
#define _IMG_FILE_PNG_FORMAT            1
#define _IMG_FILE_BMP_FORMAT            2
```

VBScript

```
Const IMG_FILE_JPG_FORMAT = 0
Const IMG_FILE_PNG_FORMAT = 1
Const IMG_FILE_BMP_FORMAT = 2

Dim l_nOldImgFormat
'Получаем текущее значение
l_nOldImgFormat = l_objEventSubscription.PreferredImgFormat
'Задаем другой формат
l_objEventSubscription.PreferredImgFormat = IMG_FILE_PNG_FORMAT
```

C++

```
HRESULT CEventSubscription::get_PreferredImgFormat([out, retval] ULONG* pCurrentFormat);

HRESULT CEventSubscription::put_PreferredImgFormat([in] ULONG newFormat);
```

```
...
ULONG l_ulOldImgFormat;
l_spEventSubscription->get_PreferredImgFormat(&l_ulOldImgFormat);
// Задаем другой другой формат
l_spEventSubscription->put_PreferredImgFormat(_IMG_FILE_PNG_FORMAT);
...
```

ReplyTimeout

READ/WRITE. Целое число. Получает или задает значение таймаута, в миллисекундах, на ожидание ответа на запрос подписки на события от хоста BourneID. При создании объекта значение свойства устанавливается в 3000 (3 секунды). Минимальное значение таймаута – 1000 (1 секунда), максимальное – 10000 (10 секунд). При попытке изменить значение после установки соединения код ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст последней ошибки в "Блокировано при соединении".

VBScript

```
Dim l_nOldTimeout
'Получаем текущее значение
l_nOldTimeout = l_objEventSubscription.ReplyTimeout
'Задаем другое значение
l_objEventSubscription.ReplyTimeout = 5000
```

C++

```
HRESULT CEventSubscription::get_ReplyTimeout([out, retval] ULONG* pCurrentTimeout);

HRESULT CEventSubscription::put_ReplyTimeout([in] ULONG newTimeout);
```

```
...
ULONG l_ulOldTimeout;
l_spEventSubscription->get_ReplyTimeout(&l_ulOldTimeout);
// Задаем другое значение
l_spEventSubscription->put_ReplyTimeout(5000);
...
```

IsConnected

READ ONLY. Логическая величина. Возвращает TRUE, если соединение с хостом BourneID для получения событий открыто, и FALSE в противном случае. Соединение открывается вызовом метода Connect (описан ниже). Следует иметь ввиду, что IsConnected может вернуть FALSE, даже если соединение было успешно открыто, но в последствии было закрыто или разорвано по той или иной причине.

VBScript

```
Dim l_boolConnected : l_boolConnected = l_objEventSubscription.IsConnected
```

C++

```
HRESULT CEventSubscription::get_IsConnected([out, retval] VARIANT_BOOL* pVal);
```

```
...
VARIANT_BOOL l_vbConnected;
l_spEventSubscription->IsConnected(&l_vbConnected);
...
```

Примечание! Последующие свойства объекта относятся к информации, возвращаемой хостом BourneID в случае успешного открытия соединения. Свойства имеют атрибут READ only и могут быть запрошены после вызова метода **Connect**, закончившегося успехом.

Version

READ only. Строка. Возвращает номер версии программы BourneID, содержащийся в ответе на запрос к хосту BourneID. Формат строки: *<MajorNumber>.<MinorNumber>.<BuildNumber>*, например, "2.2.0".

VBScript

```
Dim l_sBourneIDVersion
'Получаем версию BourneID
l_sBourneIDVersion = l_objEventSubscription.Version
```

C++

```
HRESULT CEventSubscription::get_Version([out, retval] BSTR* pVersionText);
```

```
...
CComBSTR l_bstrBourneIDVersion;
l_spEventSubscription->get_Version(&l_bstrBourneIDVersion);
...
```

EditionFlag

READ only. Целое число. Зарезервировано для последующего использования. В текущей версии возвращает значение 0.

Registered

READ only. Логическая величина. Возвращает TRUE, если активация программы BourneID была выполнена, и FALSE, в противном случае.

VBScript

```
Dim l_boolBourneIDRegistered
l_boolBourneIDRegistered = l_objEventSubscription.Registered
```

C++

```
HRESULT CEventSubscription::get_Registered([out, retval] VARIANT_BOOL* pIsRegistered);
```

```
...
VARIANT_BOOL l_vbBourneIDRegistered;
l_spEventSubscription->get_Registered(&l_vbBourneIDRegistered);
...
```

LicExpired

READ only. Логическая величина. Возвращает TRUE, если активация программа BourneID не выполнена и срок пробного использования истек, либо если активация была выполнена с ограничением по дате (временная лицензия), и срок временной лицензии истек. Возвращает FALSE во всех других случаях.

VBScript

```
Dim l_boolLicExpired : l_boolLicExpired = l_objEventSubscription.LicExpired
```

C++

```
HRESULT CEventSubscription::get_LicExpired([out, retval] VARIANT_BOOL* pIsLicExpired);
```

```
...
VARIANT_BOOL l_vbLicExpired;
l_spEventSubscription->get_LicExpired(&l_vbLicExpired);
...
```

Методы объекта EventSubscription

Методы SetCallbackNull, SetCallbackEvent и SetCallbackWindow задают один из механизмов обратного вызова кода вашего приложения, при получении события от хоста BourneID после установки соединения. Принятые от BourneID события помещаются в очередь. При создании объекта EventSubscription механизм обратного вызова установлен в значение "Отсутствует". Это означает, что ваше приложение никак не оповещается о приеме событий. В этом случае, получить принятое событие из очереди возможно, только периодически запрашивая событие объекта. Если в очереди есть принятые события, то самое "старое" событие будет передано в код приложения. Если событий в очереди нет, то будет возвращена ошибка.

SetCallbackNull

Задает отсутствие механизма обратного вызова при получении события от хоста BourneID. Механизм обратного вызова может быть изменен, только если соединение с хостом в данный момент не установлено. В противном случае метод возвратит **S_FALSE**, код последней ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст ошибки в "Блокировано при соединении". После установки соединения код приложения должен периодически запрашивать у объекта EventSubscription принятое событие.

Ниже приведен пример кода условного приложения, получающего событий при отсутствии механизма обратного вызова. Некоторые методы и объекты, используемые в примере, будут документированы далее.

VBScript

```
l_objEventSubscription.SetCallbackNull   'Нет механизма обратного вызова

'Соединяемся
l_objEventSubscription.Connect 127, 7

Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    Dim l_objBIDEvent

    Do
        Set l_objBIDEvent = Nothing

        Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

        If l_objEventSubscription.LastErrorCode <> 0 Then
            'Нет событий в очереди. Ждем некоторое время
            WScript.Sleep 1000
            'Можем также выполнять другие проверки на выход из цикла
        Else
            'Событие из очереди получено
            Dim l_nBIDEventType
            l_nBIDEventType = l_objBIDEvent.Type
            Select Case l_nBIDEventType
                Case 0 Exit Do
                   ‘ Соединение закрыто хостом BourneID или по другой причине
                Case Else
```

```
                    'Обрабатываем событие
                    ...
                End Select
            End If
        Loop

        Set l_objBIDEvent = Nothing
        l_objEventSubscription.Disconnect
End If
```

C++

```
HRESULT CEventSubscription::SetCallbackNull();
```

```
...
l_spEventSubscription->SetCallbackNull();

HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);

if (hr == S_OK)
{
    // Соединились успешно
    BOOL forever = TRUE;

    while(forever) {
        CComPtr<IBIDEvent> l_spBIDEvent;

        hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);

        if (hr != S_OK) {
            // Нет событий в очереди. Ждем некоторое время.
            Sleep(1000);
        }
        else {
            UINT uEventType;
            l_spBIDEvent->get_Type(&uEventType);

            switch (uEventType) {
                case _BID_EVENT_TYPE_CONN_CLOSED:
                    // Соединение закрыто. Можно запросить у события причину закрытия.
                    ...
                    forever = FALSE;
                    break;

                default:
                    // Обрабатываем событие
                    ...
                    break;
            }
        }
    }

    l_spEventSubscription->Disconnect();
}
...
```

SetCallbackEvent

Задает механизм обратного вызова через установку Event в состояние signaled при получении события от хоста BourneID. Метод создает Event и возвращает его вызывающей программе. После установки соединения код приложения вызывает функции синхронизации WaitForSingleObject/WaitForMultipleObject для ожидания события на Event. Созданный в методе Event не должен удаляться приложением по окончании использования. Event объект удаляется автоматически. Ожидание события на Event может выполняться как в основном потоке приложения, так и в отдельном потоке, который должен обязательно инициализировать библиотеку OLE через вызов CoInitialize(NULL). Механизм обратного вызова может быть задан, только если соединение с хостом в данный момент не установлено. В противном случае метод возвратит **S_FALSE**, код последней ошибки будет установлен в `ERROR_CONTENT_BLOCKED`, а текст ошибки в "Блокировано при соединении".

Ниже приведен пример кода условного приложения, получающего события BourneID через Event. Некоторые методы и объекты, используемые в примере, будут документированы далее.

VBScript

```
Dim l_hEvent  'Объявляем переменную в которую метод занесет Event

l_objEventSubscription.SetCallbackEvent l_hEvent

'Соединяемся
l_objEventSubscription.Connect 127, 7

Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    Dim l_objBIDEvent

    Do
        Dim l_nRetValue

        'Ждем 1 секунду на Event
        l_nRetValue = l_objEventSubscription.WaitForCallbackEvent l_hEvent, 1000

        If l_nRetValue = 0 Then    'Есть событие в очереди
            Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

            'Событие из очереди забрано
            Dim l_nBIDEventType
            l_nBIDEventType = l_objBIDEvent.Type
            Select Case l_nBIDEventType
                Case 0 Exit Do
                   'Соединение закрыто хостом BourneID или по другой причине
                Case Else
                    'Обрабатываем событие
                    ...
            End Select
            Set l_objBIDEvent = Nothing
        End If
    Loop

    Set l_objBIDEvent = Nothing
```

```
    l_objEventSubscription.Disconnect
End If
```

C++

```
HRESULT CEventSubscription::SetCallbackEvent([out, retval] ULONG *pEvent);
```

```
...
HANDLE hEvent = NULL;
l_spEventSubscription->SetCallbackEvent((ULONG*)&hEvent);

HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);

if (hr == S_OK)
{
    // Соединились успешно
    BOOL forever = TRUE;

    while(forever) {
        // Ждем 1 секунду на Event
        if (WAIT_OBJECT_0 == WaitForSingleObject(hEvent, 1000))
        {
            // Есть событие в очереди
            CComPtr<IBIDEvent> l_spBIDEvent;

            // забираем событие из очереди
            spEventSubscription->GetBIDEvent(&l_spBIDEvent);

            UINT uEventType;
            l_spBIDEvent->get_Type(&uEventType);

            switch (uEventType) {
                case _BID_EVENT_TYPE_CONN_CLOSED:
                    // Соединение закрыто. Можно запросить у события причину закрытия соединения
                    ...
                    forever = FALSE;
                    break;
                default:
                    // Обрабатываем событие
                    ...
                    break;
            }
        }
    }

    l_spEventSubscription->Disconnect();
}
...
```

SetCallbackWindow

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

Задает механизм обратного вызова через посылку сообщения окну программы. В параметрах метода задается handle окна и номер сообщения. После установки соединения, при поступлении события в очередь, оконная функция приложения получает сообщение с заданным номером. Далее приложение может забрать

сообщение из очереди. Механизм обратного вызова может быть изменен, только если соединение с хостом в данный момент не установлено. В противном случае метод возвратит **S_FALSE**, код последней ошибки будет установлен в ERROR_CONTENT_BLOCKED, а текст ошибки в "Блокировано при соединении". Рекомендуется задавать номер сообщения не менее чем WM_USER+100.
При получении сообщения функцией окна wParam содержит тип сообщения, занесенного в очередь.

Ниже приведен пример кода на C++ условного приложения, получающего события BourneID через вызов функции окна. Некоторые методы и объекты, используемые в примере, будут документированы далее.

C++

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

HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);
...
...

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

        case WM_USER+200:
        {
            BOOL forever = TRUE;

            while(forever) {
                CComPtr<IBIDEvent> l_spBIDEvent;

                // пытаемся забрать событие из очереди
                HRESULT hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);

                if (hr == S_OK) {
                    // событие из очереди получено
                    ...
                }
                else {
                    // событие не получено
                    DWORD dwLastErrorCode;
                    spEventSubscription->get_LastErrorCode(&dwLastErrorCode);

                    switch (dwLastErrorCode) {
                        case ERROR_NO_MORE_ITEMS:
                            // Нормально. Все события из очереди забрали.
                            forever = FALSE;
                            break;
                        default:
                            // Ошибка при создании объекта BIDEvent. Такого не должно быть,
                            // если только при большой нехватке ресурсов в системе.
                            forever = FALSE;
                            break;
                    }
                    break;
                }
            }
            return 0;
        }
        ...
        ...
    }
    return DefWindowProc(hwnd, msg, wParam, lParam);
}
```

Connect

```
HRESULT CEventSubscription::Connect([in] ULONG uEventMask, [in] ULONG uListMask);
```

Открывает соединение с хостом BourneID, выполняет запрос на подписку на события, и ожидает ответа с подтверждением.

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства **LastErrorCode**, а тест ошибки - **LastErrorText**.

uEventMask

Числовой параметр, определяющий, события каких типов будет посылать хост BourneID.
Значение параметра может иметь любую комбинацию значений:
_EF_CAMERA_STATUS_CHANGED – Изменился статуса камеры;
_EF_NEW_FACE_DETECTED – Новый объект появился в камере;
_EF_PERSON_RECOGNIZED_PRIMARY – Персона распознана первично;
_EF_PERSON_RECOGNIZED_MORE_CONFIDENT – Персона распознана с большей уверенностью;
_EF_ANOTHER_PERSON_INSTEAD_PREVIOUS – Распознана другая персона вместо предыдущей;
_EF_RECOGNIZED_PERSON_HAS_GONE – Распознанная персона пропала из кадра;
_EF_FACE_HAS_GONE – Нераспознанный объект пропал из кадра;
_EF_ALL – События всех типов;
_EF_PERSON_RECOGNIZED_MASK – События, связанные с распознаванием персон.

Константы определены в файле BourneIdMgmt_def.h

```
#define _EF_CAMERA_STATUS_CHANGED               0x00000001
#define _EF_NEW_FACE_DETECTED                   0x00000002
#define _EF_PERSON_RECOGNIZED_PRIMARY           0x00000004
#define _EF_PERSON_RECOGNIZED_MORE_CONFIDENT    0x00000008
#define _EF_ANOTHER_PERSON_INSTEAD_PREVIOUS     0x00000010
#define _EF_RECOGNIZED_PERSON_HAS_GONE          0x00000020
#define _EF_FACE_HAS_GONE                       0x00000040
#define _EF_ALL      (_EF_CAMERA_STATUS_CHANGED | _EF_NEW_FACE_DETECTED | \
            _EF_PERSON_RECOGNIZED_PRIMARY | _EF_PERSON_RECOGNIZED_MORE_CONFIDENT | \
            _EF_ANOTHER_PERSON_INSTEAD_PREVIOUS | _EF_RECOGNIZED_PERSON_HAS_GONE | \
            _EF_FACE_HAS_GONE)
#define _EF_PERSON_RECOGNIZED_MASK     (_EF_PERSON_RECOGNIZED_PRIMARY | \
             _EF_PERSON_RECOGNIZED_MORE_CONFIDENT | _EF_ANOTHER_PERSON_INSTEAD_PREVIOUS | \
             _EF_RECOGNIZED_PERSON_HAS_GONE)
```

uListMask

Числовой параметр, определяющий, для каких списков учета персон, события, имеющие параметры распознанной персоны будут посылаться хостом BourneID. Значение параметра может иметь любую комбинацию значений:
_LF_GREEN_LIST – Зеленый список;
_LF_RED_LIST – Красный список;
_LF_GRAY_LIST – Серый список;
_LF_ALL_LISTS – Все списки.

Если будет задано значение 0, то оно будет заменено на _LF_ALL_LISTS.

Константы определены в файле BourneIdMgmt_def.h

```
#define _LF_GREEN_LIST            0x00000001
#define _LF_RED_LIST              0x00000002
```

```
#define _LF_GRAY_LIST              0x00000004
#define _LF_ALL_LISTS              (_LF_GREEN_LIST | _LF_RED_LIST | _LF_GRAY_LIST)
```

При выполнении метода используются значения следующих свойств объекта, заданных явно, или со значениями по умолчанию:

- **ServerHost** – Имя или IP-адрес компьютера с программой BourneID. По умолчанию *localhost*;
- **ServerPort** – Номер порта TCP, по которому хост BourneID ожидает запросы. По умолчанию *21077*;
- **ServerPassword** – Пароль для запроса к хосту BourneID. По умолчанию *пустой пароль*;
- **PreferredImgFormat** – Сообщает хосту BourneID, в каком формате нужно возвращать графические изображения. По умолчанию *JPG;*
- **ReplyTimeout** – Время ожидания соединения и ответа от хоста BourneID. По умолчанию *3000 мсек.*

При выполнении метода, компонент:

- Открывает TCP соединение;
- Передает пакет запроса на подписку;
- Ожидает ответа от хоста BourneID;
- Заполняет значения свойств объекта EventSubscription, связанные с возвращаемой информацией;

Если все стадии завершаются успешно, то метод возвращает значение **S_OK** (0). Соединение остается открытым и передаваемые хостом события помещаются в очередь объекта. Код приложения может запрашивать значения полученных свойств, а также получать поступающие в очередь события.
Далее соединение может быть закрыто:

- Вызовом метода Disconnect();
- По инициативе хоста BourneID, например, при завершении работы программы;
- При разрыве TCP соединения по различным причинам.

  В двух последних случаях в очередь принятых сообщений объекта автоматически будет помещено событие с типом «Соединение закрыто».

Если при выполнении метода Connect на любой стадии фиксируется ошибка, то дальнейшей выполнение прекращается, соединение закрывается и возвращается значение **S_FALSE** (1). Код ошибки, а также текстовое описание причины ошибки можно извлечь из значений свойств **LastErrorCode** и **LastErrorText**.

Код ошибки	Описание причины ошибки
1	"Неправильный пароль" "Ошибка создания XML содержимого запроса"
2	"Неправильная структура XML запроса. Отсутствует тег <Type>."
3	"Неподдерживаемый тип запроса" "Неверно задан параметр запрашиваемого сервиса (ServiceType)" "Неверно задан формат графических файлов (PreferredImgFormat)"
8	"Ошибка выделения памяти в системе"
87	"Неверно задан параметр маски событий" "Неверно задан параметр запрашиваемого сервиса" "Невозможно определить IP-адрес хоста"

101	"Неизвестная ошибка select() на ожидании входящих данных." "Неизвестная ошибка при разборе XML ответа" "Ошибка при разборе XML ответа: <........>"
102	"Непонятный "мусор" в ответе на запрос." "Размер данных ответа на запрос слишком велик." "Тело ответа не является XML документом допустимого формата" "XML ответа не содержит корневой элемент" "Корневой элемент XML ответа имеет неправильное имя" "Отсутствует тег <ErrorCode> в XML ответа" "Отсутствует тег <BourneID> в XML ответа" "Отсутствует тег <Camera> в XML ответа"
103	"Ошибка сохранения графического файла с кадром камеры"
1230	"Соединение уже установлено"
1232	"Невозможно определить IP-адрес хоста"
10000	"BourneID не ответил на запрос в заданное время"
От 10004 до 11004	Ошибки WinSocket "Открытие сокета: ...." "Привязка сокета: ...." "Задание размера буфера приема сокета: ...." "Задание размера буфера передачи сокета: ...." "Перевод сокета в Blocking режим: ...." "Установка соединения с хостом BourneID: ...." "Посылка запроса на хост BourneID: ...." "Ожидание входящих данных в ответ на запрос: ...." "Чтение данных ответа на запрос: ...."
Другие коды	Описание ошибки можно получить из значения свойства **LastErrorText**

VBScript

```
'Соединяемся и подписываемся на все типы событий всех списков учета
l_objEventSubscription.Connect 127, 7

Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    ...
Else
    'Получить описание ошибки.
    Dim l_sLastErrorText
    l_sLastErrorText = l_objEventSubscription.LastErrorText
    ...
End If
```

C++

```
...
HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);
```

```
if (hr == S_OK) {
    // Соединились успешно.
    ...
}
else {
    // Получить описание ошибки.
    CComBSTR l_bstrErrorText;
    l_spEventSubscription->get_LastErrorText(&l_bstrErrorText);
    ...
}
```

Disconnect

```
HRESULT CEventSubscription::Disconnect();
```

Закрывает соединение с хостом BourneID, открытое методом Connect().

- Возвращает **S_OK** в случае успеха и **S_FALSE** в случае ошибки. В случае ошибки, код ошибки может быть возвращен последующим запросом значения свойства **LastErrorCode**, а тест ошибки - **LastErrorText**.

Код ошибки	Описание причины ошибки
2250	"Соединение не установлено"
Другие коды	Описание ошибки можно получить из значения свойства **LastErrorText**

Примечание! После успешного открытия соединения методом **Connect()** всегда вызывайте метод **Disconnect()**, даже если соединение было разорвано по инициативе хоста BourneID или при разрыве сети.

VBScript

```
'Соединяемся
l_objEventSubscription.Connect 127, 7

Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    ...
    l_objEventSubscription.Disconnect
End If
```

C++

```
...
HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);
if (hr == S_OK) {
    // Соединились успешно.
    ...
    hr = l_spEventSubscription->Disconnect();
}
```

WaitForCallbackEvent

```
HRESULT CEventSubscription::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. Если для объекта EventSubscription задается механизм обратного вызова с использованием Event (**SetCallbackEvent**), то метод WaitForCallbackEvent представляет собой полный эквивалент WaitForSingleObject.

hEvent

Event объект, возвращаемые при вызове метода **SetCallbackEvent**;

lMilliseconds

Таймаут ожидания события, в миллисекундах. Значение таймаута может варьироваться от 0 (поверка состояния и возврат) до INFINITE (-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

```
Dim l_hEvent  'Объявляем переменную в которую метод SetCallbackEvent занесет Event

'Задаем для объекта Eventsubscription механиз обратного вызова с использованием Event
l_objEventSubscription.SetCallbackEvent l_hEvent

'Соединяемся с хостом BourneID и подписываемся на все события для всех списков учета
l_objEventSubscription.Connect 127, 7

'Удалось соединиться?
Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    Dim l_objBIDEvent

    Do
        Dim l_nRetValue
```

```
        'Ждем 1 секунду на Event
        l_nRetValue = l_objEventSubscription.WaitForCallbackEvent l_hEvent, 1000

        If l_nRetValue = 0 Then    'Есть событие в очереди
            'Забираем событие из очереди
            Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

            'Получаем тип события BourneID
            Dim l_nBIDEventType
            l_nBIDEventType = l_objBIDEvent.Type

            Select Case l_nBIDEventType
                Case 0 Exit Do
                   'Соединение закрыто хостом BourneID или по другой причине
                   'Событий больше не будет до нового соединения
                Case Else
                    'Обрабатываем событие
                    ...
            End Select
            Set l_objBIDEvent = Nothing
        End If
    Loop

    Set l_objBIDEvent = Nothing
    l_objEventSubscription.Disconnect
End If
```

ClearEventList

```
HRESULT CEventSubscription::ClearEventList();
```

Чистить внутреннюю очередь принятых от хоста BourneID событий. Внутренняя очередь может содержать максимально 256 событий, принятых от хоста BourneID. Если приложение не будет забирать из очереди события, то при достижении максимума числа событий в очереди, новые события будут игнорироваться. Поэтому ваше приложение должно «забирать» события из очереди – вызывать в цикле метод **GetBIDEvent** до тех пор, пока этот метод не вернет пустой объект события.

После закрытия или разрыва соединения, во внутренней очереди могут остаться события, не забранные приложением. Если эти события не представляют интереса, то вы можете очистить очередь, вызвав метод ClearEventList.

Вызов метода не является обязательным условием для повторного открытия соединения, так как метод Connect очищает очередь перед соединением с хостом BourneID.

GetBIDEvent

```
HRESULT CEventSubscription::GetBIDEvent([out, retval] IBIDEvent** ppBIDEvent);
```

Создает объект **BIDEvent**, забирая из внутренней очереди объекта EventSubscriprion самое старое событие, принятое от хоста BourneID и возвращает приложению интерфейс на объект BIDEvent. Событие из внутренней очереди удаляется.

- Возвращает **S_OK** в случае успеха и **S_FALSE** или **E_FAIL** в случае ошибки.

ppBIDEvent

Адрес указателя на интерфейс создаваемого объекта BIDEvent. При выходе указатель будет заполнен значением в случае успеха. В случае ошибки, содержимое по адресу указателя останется неизменным.

В случае ошибки, код ошибки может быть получен запросом значения свойства **LastErrorCode**, а текст ошибки - значения свойства **LastErrorText**:

Возврат GetBIDEvent	LastErrorCode	Значение	LastErrorText. Описание
S_OK	ERROR_SUCCESS	0	"". Объект BIDEvent успешно создан.
S_FALSE	ERROR_NO_MORE_ITEMS	259	"Нет событий в списке". Нормальная ситуация. Приложение забрало все события из очереди. Очередь пуста.
E_FAIL	ERROR_OBJECT_NOT_FOUND	4312	"Ошибка создания объекта BIDEvent". Нехватка ресурсов в системе.

Полученный от метода объект (интерфейс объекта) по окончании его использования **должен быть удален в коде внешнего ПО**.

VBScript

```
l_objEventSubscription.SetCallbackNull   'Нет механизма обратного вызова

'Соединяемся
l_objEventSubscription.Connect 127, 7

Dim l_nLastError
l_nLastError = l_objEventSubscription.LastErrorCode

If l_nLastError = 0 Then
    'Соединились успешно.
    Dim l_objBIDEvent

    Do
        Set l_objBIDEvent = Nothing
        Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent
        If l_objEventSubscription.LastErrorCode <> 0 Then
            'Нет событий в очереди. Ждем некоторое время
            WScript.Sleep 1000
            'Можем также выполнять другие проверки на выход из цикла
        Else
            'Событие из очереди получено
            Dim l_nBIDEventType
            l_nBIDEventType = l_objBIDEvent.Type
            Select Case l_nBIDEventType
                Case 0 Exit Do
                   'Соединение закрыто хостом BourneID или по другой причине
                Case Else
                    'Обрабатываем событие
                    ...
            End Select
        End If
    Loop
```

```
    Set l_objBIDEvent = Nothing
    l_objEventSubscription.Disconnect
End If
```

C++

```
...

l_spEventSubscription->SetCallbackNull();

HRESULT hr = l_spEventSubscription->Connect(_EF_ALL, _LF_ALL_LISTS);

if (hr == S_OK)
{
    // Соединились успешно
    BOOL forever = TRUE;

    while(forever) {
        CComPtr<IBIDEvent> l_spBIDEvent;

        hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);

        if (hr == S_OK) {
            UINT uEventType;
            l_spBIDEvent->get_Type(&uEventType);

            switch (uEventType) {
                case _BID_EVENT_TYPE_CONN_CLOSED:
                    // Соединение закрыто. Можно запросить у события причину закрытия.
                    ...
                    forever = FALSE;
                    break;

                default:
                    // Обрабатываем событие
                    ...
                    break;
            }
        }
        else {
            // Нет событий в очереди.
            Sleep(1000);
        }


        // Объект BIDEvent будет удален автоматически, так как l_spBIDEvent это SMART указатель
        // на интерфейс объекта.
        // Если используется обычный указатель вида
        //
        // IBIDEvent* l_pBIDEvent = NULL;
        // hr = spEventSubscription->GetBIDEvent(&l_pBIDEvent);
        // то необходимо использовать Release для вызова деструктора объекта
        // l_pBIDEvent.Release();

    }
    l_spEventSubscription->Disconnect();
}
...
```

Объект BIDEvent

Объект, хранящий все параметры события переданного хостом BourneID. Объект имеет только свойства, и не имеет методов. Объект имеет единственный интерфейс **IBIDEvent**. Все свойства объекта имеют атрибуты READ ONLY.

Использование объектов в коде приложения

Объекты BIDEvent создаются при вызове метода **GetBIDEvent** объекта **EventSubscription**. Метод возвращает интерфейс на созданный объект. По окончании использования объекта, его необходимо удалить. Для автоматического удаления объекта BIDEvent можно использовать smart указатель на интерфейс или явно вызывать Release у наследованного интерфейса IUnknown.

Общие свойства объекта BIDEvent

Type

Целое число. **Тип события** BourneID. Может принимать значения:

0	_BID_EVENT_TYPE_CONN_CLOSED	Соединение было закрыто
1	_BID_EVENT_TYPE_CAMERA_STATUS	Изменился статус или текущая операция камеры
2	_BID_EVENT_TYPE_NEW_FACE_DETECTED	Новый объект появился в камере
3	_BID_EVENT_TYPE_RECOGNIZED_PRIMARY	Персона распознана первично
4	_BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT	Персона распознана с большей уверенностью
5	_BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS	Распознана другая персона вместо предыдущей
6	_BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE	Распознанная персона пропала из кадра
7	_BID_EVENT_TYPE_FACE_HAS_GONE	Нераспознанный объект пропал из кадра

VBScript

```
...
Dim l_objBIDEvent
Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

If l_objEventSubscription.LastErrorCode = 0 Then
    'Событие из очереди получено. Какой тип события?
    Dim l_nBIDEventType : l_nBIDEventType = l_objBIDEvent.Type
    ...
End If
```

C++

```
HRESULT CBIDEvent::get_Type([out, retval] UINT* puType);
```

```
...
CComPtr<IBIDEvent> l_spBIDEvent;
HRESULT hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);
if (hr == S_OK) {
    UINT uEventType;
    l_spBIDEvent->get_Type(&uEventType);
    ...
}
...
```

Date

Вещественное число двойной точности. Локальная **дата и время события**.

VBScript

```
...
Dim l_objBIDEvent
Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

If l_objEventSubscription.LastErrorCode = 0 Then
    'Событие из очереди получено. Какой тип события?
    Dim l_nBIDEventType : l_nBIDEventType = l_objBIDEvent.Type
    'Дата и время события?
    Dim l_dtEventDate : l_dtEventDate = l_objBIDEvent.Date
    ...
End If
```

C++

```
HRESULT CBIDEvent::get_Date([out, retval] DATE* pdtValue);
```

```
...
CComPtr<IBIDEvent> l_spBIDEvent;
HRESULT hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);
if (hr == S_OK) {
    UINT uEventType;
    l_spBIDEvent->get_Type(&uEventType);

    DATE dtEventDate;
    l_spBIDEvent->get_Date(&dtEventDate);

    // Преобразуем в SYSTEMTIME
    UDATE uDate;
    VarUdateFromDate((DATE) dtEventDate, 0, &uDate);
    ...
}
...
```

Примечание! Значения свойств **Type** и **Date** можно запрашивать у всех типов событий. Остальные свойства объекта BIDEvent можно запрашивать в зависимости от типа (значения, возвращаемого свойством **Type**).

Наборы свойств объекта BIDEvent в зависимости от типа события

Событие _BID_EVENT_TYPE_CONN_CLOSED

Если свойство **Type** имеет значение **0**, то событие содержит параметры закрытого соединения. Событие автоматически генерируется и помещается в очередь событий при закрытии соединения по инициативе хоста BourneID или разрыве сети. При закрытии соединения из кода приложения методом **Connect**, событие **_BID_EVENT_TYPE_CONN_CLOSED** не генерируется.
Приложение, получив из очереди событие **_BID_EVENT_TYPE_CONN_CLOSED**, далее может не ожидать в очереди новых событий, так как соединение закрыто.

Для этого типа события, приложение может запросить значение свойств:

ConnClosedErrorCode	Код закрытия соединения. Если код равен 0, то соединение было закрыто по инициативе хоста BourneID. В противном случае соединение было разорвано по другой причине.
`ConnClosedErrorText`	Содержит текстовое описание причины разрыва соединения.

Событие _BID_EVENT_TYPE_CAMERA_STATUS

Свойство **Type** имеет значение **1**. Событие описывает текущее состояние или изменение статуса/текущей операции камеры. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит `_EF_CAMERA_STATUS_CHANGED`.

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры, к которой относится событие. В текущей версии всегда возвращает 0.
CameraName	Содержит имя камеры.
CameraState	Состояние камеры.
CameraOperation	Операция камеры.

Событие _BID_EVENT_TYPE_NEW_FACE

Свойство **Type** имеет значение **2**. В камере обнаружено лицо человека (новый объект), которому присваивается номер **KeyNumber**. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит `_EF_NEW_FACE_DETECTED`.

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры.
CameraName	Содержит имя камеры.
KeyNumber	Номер нового объекта.
ImageFile	Графический файл, с прямоугольником лица нового объекта.

Событие _BID_EVENT_TYPE_RECOGNIZED_PRIMARY

Свойство **Type** имеет значение **3**. Для нового объекта в камере произошло распознавание одной из персон с уровнем уверенного распознавания. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит `_EF_PERSON_RECOGNIZED_PRIMARY`, а параметр **uListMask** бит соответствующего списка учета, к которому принадлежит распознанная персона.

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры.
CameraName	Содержит имя камеры.
KeyNumber	Номер объекта.
ImageFile	Графический файл, с прямоугольником лица персоны в кадре.
PersonID	Идентификатор персоны.
Confidence	Уверенность распознавания.
ListNum	Список учета персоны.
PersonSurname	Фамилия.
PersonForename	Имя.
PersonSecondname	Отчество.
PersonAlias	Псевдоним.
PersonExtKey	Внешний ключ персоны.
Gender	Гендер.
DateOfBirth	Дата рождения.
RepeatCount	Число распознаваний с момента появления объекта в камере.

Событие _EF_PERSON_RECOGNIZED_MORE_CONFIDENT

Свойство **Type** имеет значение **4**. Персона распознана с большей уверенностью. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит `_EF_PERSON_RECOGNIZED_MORE_CONFIDENT`, а параметр **uListMask** бит соответствующего списка учета, к которому принадлежит распознанная персона. Событие посылается хостом BourneID, если при текущей попытке распознавания значение уверенности распознавания **Confidence** превысит значение при первичном или предшествующем распознавании персоны.

Для этого типа событий, приложение может запрашивать те же свойства, что и для события `_BID_EVENT_TYPE_RECOGNIZED_PRIMARY`.

Событие _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS

Свойство **Type** имеет значение **5**. Для объекта, имеющего номер KeyNumber произошло распознавание персоны, отличной от распознанной ранее. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит

_EF_ANOTHER_PERSON_INSTEAD_PREVIOUS, а параметр **uListMask** бит соответствующего списка учета, к которому принадлежит распознанная ранее или текущая персона.

Распознавание другой персоны вместо предыдущей может происходить по двум причинам:

1. В программе BourneID задан низкий порог уверенного распознавания. В этом случае, при первичном распознавании, при «неудачном» положении лица нового объекта программа может распознать одну персону, а при последующих попытках – другую;
2. В камере «двигаются» несколько объектов, со значительной скоростью изменения координат лиц. Если в настройках программы BourneID включена опция **"Допускать короткую потерю области лица"**, то возможна ситуация, когда область лица первого объект уйдет из камеры, а ее место быстро займет область лица другого объекта.

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры.
CameraName	Содержит имя камеры.
KeyNumber	Номер объекта.
ImageFile	Графический файл, с прямоугольником лица персоны.
PersonID	Идентификатор текущей распознанной персоны.
Confidence	Уверенность распознавания текущей персоны.
ListNum	Список учета текущей персоны.
PersonSurname	Фамилия.
PersonForename	Имя.
PersonSecondname	Отчество.
PersonAlias	Псевдоним.
PersonExtKey	Внешний ключ персоны.
Gender	Гендер.
DateOfBirth	Дата рождения.
RepeatCount	Число распознаваний текущей персоны.
OldPersonID	Идентификатор ранее распознанной персоны для этого объекта.
OldConfidence	Уверенность распознавания ранее распознанной персоны.
OldListNum	Список учета ранее распознанной персоны.

Событие _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE

Свойство **Type** имеет значение **6**. Персона, которая была распознана, пропала из кадра камеры. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит _EF_RECOGNIZED_PERSON_HAS_GONE, а параметр **uListMask** бит соответствующего списка учета, к которому принадлежит распознанная персона.
Если в настройках программы BourneID опция **"Допускать короткую потерю области лица"** выключена, то событие будет посылаться при потере области лица объекта в камере. Если опция включена, то событие будет посылаться, если лицо объект не появится в течение короткого интервала (около половины секунды).

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры.
CameraName	Содержит имя камеры.
KeyNumber	Номер объекта.
PersonID	Идентификатор распознанной персоны.
ListNum	Список учета персоны.
PresenceSeconds	Длительность нахождения распознанной персоны в кадре камеры.

Событие _BID_EVENT_TYPE_FACE_HAS_GONE

Свойство **Type** имеет значение **7**. Нераспознанный объект пропал из кадра камеры. Этот тип событий будет посылаться хостом BourneID, если при установке соединения в параметре **uEventMask** метода **Connect** будет присутствовать бит `_EF_FACE_HAS_GONE`.

Для этого типа событий, приложение может запрашивать значения свойств:

CameraNumber	Номер камеры.
CameraName	Содержит имя камеры.
KeyNumber	Номер объекта.
PresenceSeconds	Длительность нахождения объекта в кадре камеры.

Свойства объекта BIDEvent, зависящие от типа события

ConnClosedErrorCode

Целое число. **Код ошибки закрытия соединения** с хостом BourneID.
Значение **0** соответствует закрытию соединения по инициативе хоста BourneID, например при завершении работы программы или при выключении опции **"Включить Management Server "** в настройках программы с открытым соединением. Код ошибки отличный от 0 свидетельствует о разрыве соединения по различным причинам, например разрыве сети. Как правило, числовой код ошибки лежит в диапазоне значений от 10004 до 11004, что соответствует ошибкам работы с Socket. Текст описания ошибки может быть получен запросом значения свойства **ConnClosedErrorText**.

Значение свойства может быть получено для события типа `_BID_EVENT_TYPE_CONN_CLOSED (0)`.

VBScript

```
...
Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

Select Case l_objBIDEvent.Type
    Case 0   'Соединение закрыто
        'Получаем код ошибки закрытия соединения
```

```
        Dim l_nErrorCode : l_nErrorCode = l_objBIDEvent.ConnClosedErrorCode
        If l_nErrorCode = 0 Then
            'Соединение закрыто хостом BourneID
            ...
        Else
            'Соединение разорвано
            Dim l_sErrorText : l_sErrorText = l_objBIDEvent.ConnClosedErrorText
            ...
        End If
        ...
End Select
```

C++

```
HRESULT CBIDEvent::get_ConnClosedErrorCode([out, retval] UINT* pErrorCode);
```

```
...
CComPtr<IBIDEvent> l_spBIDEvent;
HRESULT hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);
if (hr == S_OK) {
    UINT uEventType;
    l_spBIDEvent->get_Type(&uEventType);

    switch (uEventType) {
        case _BID_EVENT_TYPE_CONN_CLOSED: // Соединение закрыто
        {
            UINT uErrorCode;
            l_spBIDEvent->get_ConnClosedErrorCode(&uErrorCode);

            if (uErrorCode == 0) {
                // Соединение закрыто хостом BourneID
                ...
            }
            else  {
                // Соединение разорвано
                CComBSTR bstrErrorText;
                l_spBIDEvent->get_ConnClosedErrorText(&bstrErrorText);
                ...
            }
            break;
        }
        ...
    }
}
...
```

ConnClosedErrorText

Строка. Содержит описание **причины закрытия соединения**.

Значение свойства может быть получено для события типа `_BID_EVENT_TYPE_CONN_CLOSED (0)`.

C++

```
HRESULT CBIDEvent::get_ConnClosedErrorText([out, retval] BSTR* pErrorText);
```

CameraNumber

Целое число. В текущей версии значение свойства **всегда 0**. Значение свойства может быть получено для событий всех типов, за исключением `_BID_EVENT_TYPE_CONN_CLOSED (0)`.

VBScript

```
...
Dim l_nCameraNumber : l_nCameraNumber = l_objBIDEvent. CameraNumber
...
```

C++

```
HRESULT CBIDEvent::get_CameraNumber([out, retval] UINT* pVal);
```

```
...
UINT uCameraNumber;
l_spBIDEvent->get_CameraNumber(&uCameraNumber);
...
```

CameraName

Строка. Содержит **имя текущей камеры**. Если в настройках программы BourneID камеры не заданы, или текущая камера в программе не выбрана, то строка будет пустой. Значение свойства может быть получено для событий всех типов, за исключением `_BID_EVENT_TYPE_CONN_CLOSED (0)`.

VBScript

```
...
Dim l_sCameraNumber : l_nCameraNumber = l_objBIDEvent. CameraName
...
```

C++

```
HRESULT CBIDEvent::get_CameraName([out, retval] BSTR* pVal);
```

```
...
CComBSTR bstrCameraName;
l_spBIDEvent->get_CameraName(&bstrCameraName);
...
```

CameraState

Челое число. Содержит **статус камеры**. Может принимать значения:

- **0** (`_BID_CAMERA_STATE_IDDLE`). Камера не запущена в работу или не выбрана;
- **1** (`_BID_CAMERA_STATE_WORKS`). Камера запущена в работу.

Значение свойства может быть получено для события типа `_BID_EVENT_TYPE_CAMERA_STATUS (1)`.

VBScript

```
...
Set l_objBIDEvent = l_objEventSubscription.GetBIDEvent

Select Case l_objBIDEvent.Type
    Case 1   'Изменился статус камеры
        'Получаем статус камеры

        Dim l_nCameraState : l_nCameraState = l_objBIDEvent.CameraState
        If l_nCameraState = 0 Then
            'Камера не запущена
            ...
        Else
            'Камера работает. Дополнительно запрашиваем операцию
            Select Case l_objBIDEvent. CameraOperation
                Case 0
                    'Операция неизвестна
                    ...
                Case 1
                    'Камера стартована
                    ...
                Case 2
                    'Камера открыта
                    ...
                Case 3
                    'Камера закрыта
                    ...
                Case 4
                    'Камера «висит»
                    ...
                Case 5
                    'Камера снова работает после «зависания»
                    ...
            End Select
            ...
        End If
        ...
End Select
```

C++

```
HRESULT CBIDEvent::get_CameraState([out, retval] UINT* pVal);
```

```
...
CComPtr<IBIDEvent> l_spBIDEvent;
HRESULT hr = spEventSubscription->GetBIDEvent(&l_spBIDEvent);
if (hr == S_OK) {
    UINT uEventType;
    l_spBIDEvent->get_Type(&uEventType);

    switch (uEventType) {
        case _BID_EVENT_TYPE_CAMERA_STATUS: // Изменился статус камеры
        {
            UINT uCameraState;
            l_spBIDEvent->get_CameraState(&uCameraState);

            if (uCameraState == _BID_CAMERA_STATE_IDDLE) {
                // Камера не запущена в работу
                ...
            }
            else  {
```

```
                // Камера работает. Дополнительно запрашиваем операцию
                UINT uCameraOperation;
                l_spBIDEvent->get_CameraOperation (&uCameraOperation);
                switch (uEventType) {
                    case _BID_CAMERA_OPERATION_UNDEF: // Операция неизвестна
                        ...
                        break;
                    case _BID_CAMERA_OPERATION_START: // Камера стартована
                        ...
                        break;
                    case _BID_CAMERA_OPERATION_OPEN: // Камера открыта
                        ...
                        break;
                    case _BID_CAMERA_OPERATION_CLOSED: // Камера закрыта
                        ...
                        break;
                    case _BID_CAMERA_OPERATION_HANGUP: // Камера «висит»
                        ...
                        break;
                    case _BID_CAMERA_OPERATION_LEAVE: // Камера снова работает после «зависания»
                        ...
                        break;
                }
                ...
            }
            break;
        }
        ...
    }
}
...
```

CameraOperation

Челое число. Содержит **код операции камеры**. Может принимать значения:

- **0** (_BID_CAMERA_OPERATION_UNDEF). Операция неизвестна. Такой код операции получает приложение в событии _BID_EVENT_TYPE_CAMERA_STATUS при подключении к хосту BourneID;
- **1** (_BID_CAMERA_OPERATION_START). Камера стартовала. Состояние камеры перешло в _BID_CAMERA_STATE_WORKS;
- **2** (_BID_CAMERA_OPERATION_OPEN). Камера открыта и будет пытаться получать кадры из потока видео;
- **3** (_BID_CAMERA_OPERATION_CLOSED). Произошла ошибка при работе с камерой. Камера закрывается и через некоторое время будет произведена попытка повторного открытия камеры;
- **4** (_BID_CAMERA_OPERATION_HANGUP). Новые кадры из потока видео камеры не поступают продолжительное время. «Подвисание» характерно для работы с IP-камерами при разрыве сети;
- **5** (_BID_CAMERA_OPERATION_LEAVE). Новые кадры из потока видео снова поступают после «подвисания».

Значение свойства может быть получено для события типа _BID_EVENT_TYPE_CAMERA_STATUS (1).

C++

```
HRESULT CBIDEvent::get_CameraOperation([out, retval] UINT* pVal);
```

KeyNumber

Целое число. Содержит **номер объекта** в камере. После запуска программы BournID нумерация объектов начинается с 1. Каждый новый объект получает значение на единицу большее, чем предыдущий. Так как программа BourneID поддерживает трекинг позиции лица, то KeyNumber объекта не изменяется при перемещении объекта в пределах границ кадра камеры и даже частичного выхода за границы.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_NEW_FACE (2)
- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)
- _BID_EVENT_TYPE_FACE_HAS_GONE (7)

VBScript

```
...
Dim l_nKeyNumber : l_nKeyNumber = l_objBIDEvent.KeyNumber
...
```

C++

```
HRESULT CBIDEvent::get_KeyNumber([out, retval] UINT* pVal);
```

```
...
UINT uKeyNumber;

if (S_OK == l_spBIDEvent->get_Type(&uKeyNumber))
{
    // Номер объекта получен
    ...
}
else {
    // Запрашивали у события, не имеющего этого свойства (тип события не подходит)
    ...
}
...
```

ImageFile

Строка. Содержит полный путь к графическому файлу, в котором сохранено лицо нового объекта или распознанной персоны. Размер изображения 150x150 пикселей. Формат файла (JPG, PNG либо BMP) определяется значением свойства **PreferredImgFormat** объекта EventSubscription. Файл находится во временной папке учетной записи пользователя. Код программы не должен удалять файл с диска. Файл будет автоматически удален при удалении объекта.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_NEW_FACE (2)
- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)

VBScript

```
...
Dim l_sImageFile : l_sImageFile = l_objBIDEvent.ImageFile
...
```

C++

```
HRESULT CBIDEvent::get_ImageFile([out, retval] BSTR* pVal);
```

```
...
CComBSTR bstrImageFilePath;

if (S_OK == l_spBIDEvent->get_ImageFile(&bstrImageFilePath))
{
    // Имя файла получено
    ...
}
else {
    // Запрашивали у события, не имеющего этого свойства (тип события не подходит)
    ...
}
...
```

PersonID

Целое число. **Идентификатор** распознанной персоны в базе данных программы BourneID. Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

VBScript

```
...
Dim l_nPersonID : l_nPersonID = l_objBIDEvent.PersonID
...
```

C++

```
HRESULT CBIDEvent::get_PersonID([out, retval] UINT* pVal);
```

```
...
UINT uPersonID;

if (S_OK == l_spBIDEvent->get_PersonID(&uPersonID))
{
    // Получили ID персоны
    ...
}
...
```

Confidence

Вещественное число двойно точности (double). **Вероятность (уверенность) распознавания**, в %.
Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

VBScript

```
...
Dim l_nConfidence : l_nConfidence = l_objBIDEvent.Confidence
...
```

C++

```
HRESULT CBIDEvent::get_Confidence([out, retval] double* pVal);
```

```
...
double dblConfidence;

if (S_OK == l_spBIDEvent->get_Confidence(&dblConfidence))
{
    // Получили Confidence
    ...
}
else {
    // Запрашивали у события, не имеющего этого свойства (тип события не подходит)
    ...
}
...
```

ListNum

Целое число. **Список персоны**: Зеленый (0), Красный (1) или Серый (2).

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

VBScript

```
...
Dim l_nListNum : l_nListNum = l_objBIDEvent.ListNum
...
```

C++

```
HRESULT CBIDEvent::get_ListNum([out, retval] int* pVal);
```

```
...
int iListNum;

if (S_OK == l_spBIDEvent->get_ListNum(&iListNum))
{
    // Получили список персоны
    ...
}
else {
    // Запрашивали у события, не имеющего этого свойства (тип события не подходит)
    ...
}
...
```

PersonSurname

Строка. **Фамилия** распознанной персоны.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

VBScript

```
...
Dim l_sSurname : l_sSurname = l_objBIDEvent.PersonSurname
...
```

C++

```
HRESULT CBIDEvent::get_PersonSurname([out, retval] BSTR* pVal);
```

```
...
CComBSTR bstrSurname;

if (S_OK == l_spBIDEvent->get_PersonSurname(&bstrSurname))
{
    // Получили фамилию персоны
    ...
}
else {
    // Запрашивали у события, не имеющего этого свойства (тип события не подходит)
    ...
}
...
```

PersonForename

Строка. **Имя** распознанной персоны. Может быть пустой строкой, если в базе данных программы BourneID имя не занесено.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_PersonForename([out, retval] BSTR* pVal);
```

PersonSecondname

Строка. **Отчество** распознанной персоны. Может быть пустой строкой, если в базе данных отчество не занесено.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_PersonSecondname([out, retval] BSTR* pVal);
```

PersonAlias

Строка. **Псевдоним** распознанной персоны. Может быть пустой строкой, если в базе данных псевдоним не занесен.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_PersonAlias([out, retval] BSTR* pVal);
```

PersonExtKey

Строка. **Внешний ключ** распознанной персоны. Может быть пустой строкой, если в базе данных программы BourneID внешний ключ не занесен.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_PersonExtKey([out, retval] BSTR* pVal);
```

Gender

Целое число. **Пол персоны**: мужской (1) или женский (0).

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_Gender([out, retval] int* pVal);
```

DateOfBirth

Дата. **Дата рождения**. Если для персоны дата рождения в базе данных не задана, то значение равно 0 (нулевая дата).

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)
- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)

C++

```
HRESULT CBIDEvent::get_DateOfBirth([out, retval] DATE* pVal);
```

RepeatCount

Целое число. Счетчик **числа распознаваний персоны** с момента появления объекта в камере.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PRIMARY (3)
- _BID_EVENT_TYPE_RECOGNIZED_MORE_CONFIDENT (4)
- _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS (5)

C++

```
HRESULT CBIDEvent::get_RepeatCount([out, retval] int* pVal);
```

OldPersonID

Целое число. Идентификатор персоны распознанной ранее в этом объекте (KeyNumber). Значение свойства может быть запрошено для событий типа _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS.

C++

```
HRESULT CBIDEvent::get_OldPersonID([out, retval] UINT* pVal);
```

OldConfidence

Вещественное число двойно точности. **Вероятность (уверенность) распознавания** персоны распознанной ранее в этом объекте (KeyNumber).
Значение свойства может быть запрошено для событий типа _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS.

C++

```
HRESULT CBIDEvent::get_OldConfidence([out, retval] double* pVal);
```

OldListNum

Целое число. **Список персоны** распознанной ранее в этом объекте (KeyNumber).
Значение свойства может быть запрошено для событий типа _BID_EVENT_TYPE_ANOTHER_INSTEAD_PREVIOUS.

C++

```
HRESULT CBIDEvent::get_OldListNum([out, retval] int* pVal);
```

PresenceSeconds

Вещественное число двойно точности. **Длительность присутствия** объекта или распознанной персоны в кадре камеры, в секундах.

Значение свойства может быть запрошено для событий типа:

- _BID_EVENT_TYPE_RECOGNIZED_PERSON_HAS_GONE (6)
- _BID_EVENT_TYPE_FACE_HAS_GONE (7)

C++

```
HRESULT CBIDEvent::get_PresenceSeconds([out, retval] double* pVal);
```