Руководство по интеграции ПК Интеллект (HTTP API, IIDK, ActiveX, HTTP Сервер)

Обновлено 01.02.2021

Содержание

Скачать модуль DEMO

Примечание.

Данный модуль используется в качестве примера работы с IIDK – см. Интеграция аппаратно-программных модулей.

1 Руководство по интеграции ПК Интеллект. Введение.

В данном документе приведена информация, позволяющая обеспечить взаимодействие ПК *Интеллект* с внешними системами. ПК *Интеллект* предоставляет следующие интерфейсы для решения данной задачи:

1. Интерфейс IIDK. Используется для внедрения в систему функциональных модулей, обеспечивающих решение следующих задач:
    a. Добавление нового охранного оборудования в систему.
    b. Реализация новых сервисных функций (управление охранным оборудованием).

    Этапы интеграции модулей рассмотрены на примере демонстрационного модуля *DEMO,* исходные файлы которого приложены к документации.
    Скачать модуль *DEMO* можно на странице Руководство по интеграции ПК Интеллект.
2. Элемент управления ActiveX CamMonitor.ocx. Данный компонент является полным аналогом интерфейсного объекта **Монитор видеонаблюдения**. Он позволяет управлять камерами, просматривать архив и т.д.
3. HTTP API. Данный программный интерфейс позволяет отправлять команды и получать данные от ПК *Интеллект* при помощи HTTP-запросов.
4. RabbitMQ. Позволяет отправлять и получать сообщения в ПК *Интеллект* с помощью брокера сообщений RabbitMQ.

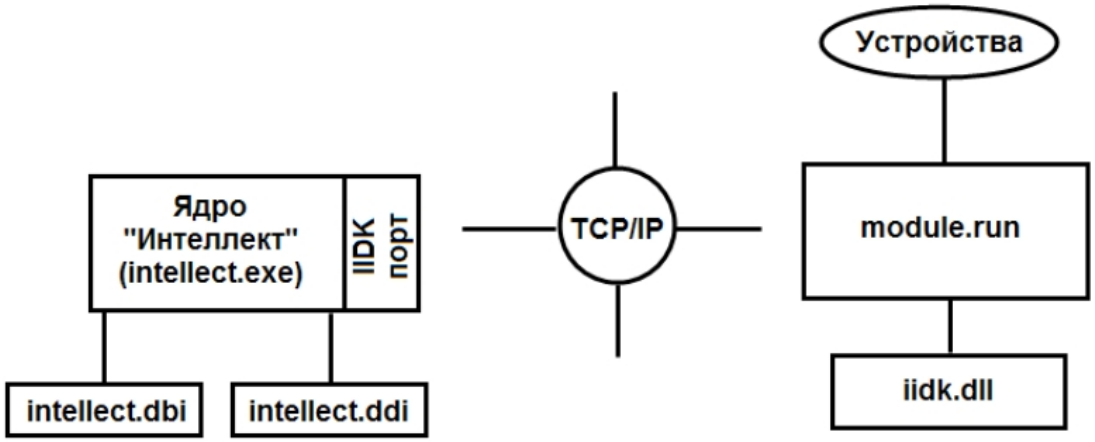
2 Интеграция аппаратно-программных модулей

2.1 INTELLECT INTEGRATION DEVELOPER KIT (IIDK)

2.1.1 Общие сведения об IIDK

2.1.1.1 Назначение IIDK

Возможность расширять систему заложена в архитектуру программного комплекса *Интеллект,* предусматривающую межзадачное взаимодействие ядра системы с функциональными модулями (смежными информационными системами) через коммуникационную среду TCP/IP. Схема взаимодействия ядра ПК *Интеллект* с внешним программным обеспечением (функциональным модулем) приведена на рисунке.



Взаимодействие ядра системы с внешним программным обеспечением выполняется посредством обмена сообщениями в коммуникационной среде, реализованного с помощью *IIDK*.

Intellect Integration Developer Kit (*IIDK*) представляет собой комплект средств разработки, используемый для интеграции охранного оборудования сторонних производителей с ПК *Интеллект*. Данный инструмент позволяет быстро и эффективно расширять систему, добавляя функциональные модули, поддерживающие новое оборудование или новые сервисные функции.

2.1.1.2 Требования к разработчику

Для использования *IIDK* требуется:

1. знание языка программирования C/C++ ;
2. знание основ программирования в Win32;
3. наличие среды разработки (*Microsoft Visual C++*, *C++ Builder, DELPHI* и др.), поддерживающей работу с dll-файлами.

> **Примечание.**
>
> Создавая lib-файл в C++ Builder 5 при помощи утилиты implib.exe, необходимо указать ключ '**- a**'.

2.1.1.3 Состав IIDK

IIDK включает в себя следующие средства разработки:

1. *iidk.ocx* – элемент управления ActiveX. Данный файл при установке ПК *Интеллект* помещается в папку Windows\System32 и регистрируется в операционной системе.
2. *ddi.exe* – программа для просмотра и редактирования DDI- и DBI- файлов. Располагается в папке <Директория установки ПК *Интеллект*>\Tools.

2.1.2 Подключение к ПК Интеллект

2.1.2.1 Параметры подключения

Взаимодействие ядра ПК *Интеллект* с функциональными модулями (смежными информационными системами) осуществляется со следующими параметрами подключения:

1. Номер порта.
   a. Для видеоподсистемы – порт 900.
   b. Для объекта **Интерфейс IIDK** – порт 1030.
   c. Для объекта **Банкомат** – порт 1009.

   > **Примечание.**
   >
   > Для подключения банкоматов можно также использовать не порт 1009 (ATM), а порт 1030 (IIDK), в таком случае объект **Банкомат** будет отображаться в дереве оборудования со значком красного креста. При этом в дереве оборудования должен быть создан объект **Интерфейс IIDK**.

2. IP-адрес компьютера, на котором функционирует ядро ПК *Интеллект.*
3. ID – идентификатор объекта подключения.

> **Внимание!**
>
> Для подключения к видеоподсистеме (порт 900) id должен быть больше 1 и не должен совпадать с id созданных в системе объектов **Интерфейс IIDK**. Для подключения к объекту **Интерфейс IIDK** (порт 1030) id равен идентификационному номеру объекта, заданному в диалоговом окне настройки ПК *Интеллект*.

> **Примечание.**
>
> Если требуется подключиться к серверу (объекту **Интерфейс IIDK**) с удаленного компьютера, не обязательно устанавливать ПК *Интеллект* на удаленный компьютер, но необходимо добавить этот компьютер в конфигурацию ПК *Интеллект* на сервере (на вкладке **Оборудование** диалогового окна **Настройка системы**), и именно на базе созданного объекта **Компьютер** создать объект **Интерфейс IIDK**. В таком случае в параметре IP функции Connect следует указывать адрес сервера, а в параметре ID идентификатор указанного объекта Интерфейс IIDK. Следует учитывать, что объект **Компьютер**, соответствующий удаленному компьютеру, будет помечен в дереве объектов красным крестом.

> **Примечание.**
>
> Если создан mdl-файл (см. раздел Разработка MDL-файла), для подключения к ядру ПК *Интеллект* объект **Интерфейс IIDK** в системе не создается. В качестве идентификатора подключения передается пустая строка, то есть id равен “”.

2.1.2.2 Объект Интерфейс IIDK

Объект **Интерфейс IIDK** позволяет управлять всеми элементами системы. Объект **Интерфейс IIDK** создается на базе объекта **Компьютер** в дереве объектов ПК *Интеллект*.

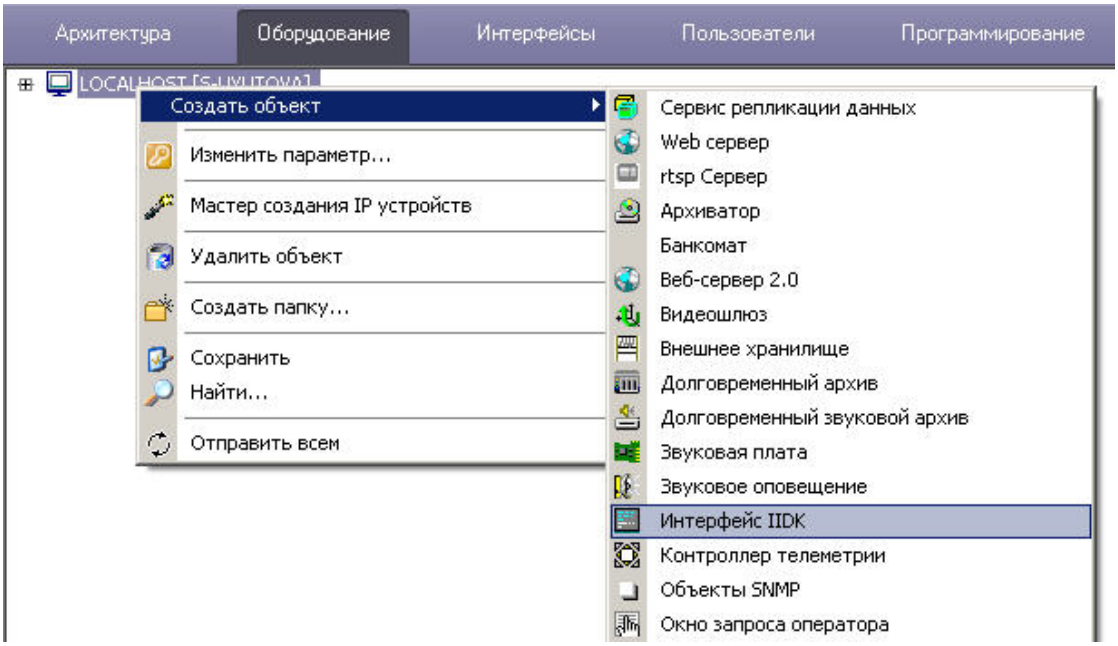
> **Примечание**
>
> Для использования объекта **Интерфейс IIDK** данная функциональная возможность должна быть разрешена в ключе активации.

Примечание

Если ПК *Интеллект* запущен в демонстрационном режиме, объект **Интерфейс IIDK** будет активирован после подключения функционального модуля к ядру системы (см. раздел Connect).

Внимание!

Идентификатор объекта **Интерфейс IIDK** не должен совпадать с идентификаторами созданных в системе объектов **Монитор**.



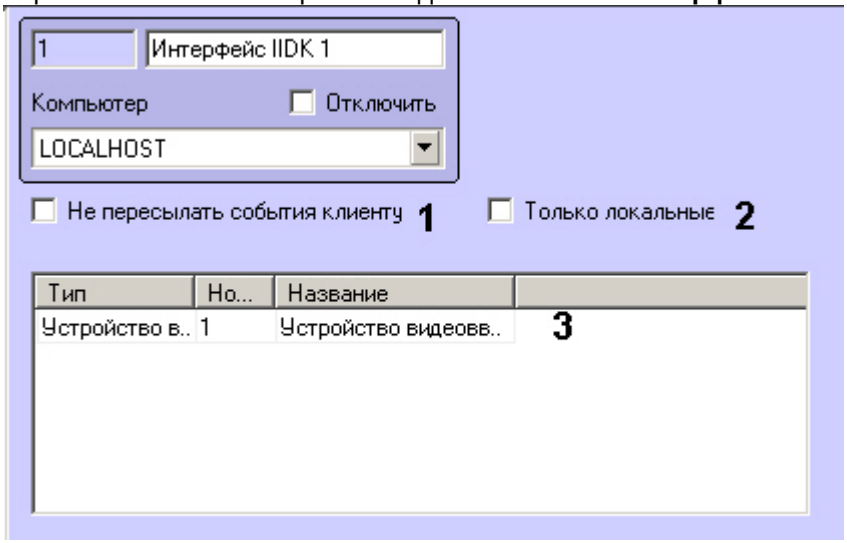
В случае использования объекта **Интерфейс IIDK** панели настройки для интегрируемых функциональных модулей (смежного программного обеспечения) не создаются.
При использовании распределенной архитектуры ПК *Интеллект* объект **Интерфейс IIDK** должен быть создан на компьютере, содержащем программное ядро, к которому выполнятся подключение. В случае, если подключение выполняется к компьютеру, на котором установлено *Рабочее место мониторинга*, то в параметрах подключения требуется указывать ip-адрес *Сервера* или *Рабочего места администратора*.

2.1.2.3 Настройка пересылки событий через объект Интерфейс IIDK

Объект **Интерфейс IIDK** позволяет настроить фильтрацию событий, передаваемых подключаемым клиентским приложениям.

Настройка фильтрации осуществляется следующим образом:

1. Перейти на панель настройки созданного объекта **Интерфейс IIDK**.



2. В случае если не требуется передавать никакие сообщения клиентским приложениям, от которых не поступают запросы к ядру, установить флажок **Не пересылать события клиенту** (1).
3. Если требуется передавать клиентским приложениям только события от объектов, созданных на базе того же объекта **Компьютер**, что и объект **Интерфейс IIDK**, установить флажок **Только локальные** (2).
4. В таблице (3) задать список объектов, события от которых следует передавать подключающимся клиентским приложениям. Для фильтрации событий от ядра необходимо указать тип CORE.

Настройка фильтрации событий завершена.

2.1.2.4 Особенности интеграции банкоматов. Объект Банкомат

Для передачи событий от ПО банкомата в ядро ПК *Интеллект* может быть использован объект **Банкомат**. Данный объект создается на базе объекта **Компьютер** на вкладке **Оборудование** диалогового окна **Настройка системы** вместо объекта **Интерфейс IIDK**.

Объект **Банкомат** позволяет отображать события банкомата ("Вставлена карта", "Отдана карта" и др.) в протоколе событий. Данные события также могут быть использованы для наложения титров на видео, настройки реакций и т.д.

> **Примечание.**
>
> Для событий, связанных с картой клиента, имеется возможность передавать маскированный номер банковской карты в параметре param0.

Список доступных для использования событий объекта **Банкомат** можно узнать при помощи утилиты ddi.exe. Работа с данной утилитой описана в документе Руководство Администратора, раздел Утилита редактирования шаблонов баз данных и файла внешних настроек ddi.exe.

Способ подключения и синтаксис сообщений для объекта **Банкомат** такие же, как для объекта **Интерфейс IIDK**, однако для отправки сообщений используется порт 1009 (см. также Параметры подключения и Синтаксис сообщений (900 порт)).

2.1.3 Функции IIDK

2.1.3.1 Connect

Для взаимодействия функционального модуля с ПК *Интеллект* необходимо выполнить подключение к ядру системы с помощью следующей функции:

```
BOOL Connect (LPCTSTR ip, LPCTSTR port, LPCTSTR id, void (_stdcall *func)(LPCTSTR msg))
```

Описание параметров функции Connect приведено в таблице.

Параметр	Описание	Пример
LPCTSTR ip	ip- адрес компьютера с ядром системы	
LPCTSTR port	порт TCP/IP, через которое происходит подключение	
LPCTSTR id	идентификатор подключения, для видео	
_stdcall *func) (LPCTSTR msg))	Callback-функция, принимающая сообщения от ПК *Интеллект*	

```
CString port = "900";

CString ip = "127.0.0.1";

CString id = "2";

BOOL IsConnect = Connect(ip, port, id, myfunc);

if (!IsConnect)

{

    // не удалось подключиться

    AfxMessageBox("Error");

}
```

Функция возвращает TRUE, если подключение выполнено, иначе - FALSE.

Все сообщения, приходящие от ядра системы, принимает Callback-функция.

Пример объявления Callback-функции:

```
void _stdcall myfunc(LPCTSTR str)

{

    printf("\r\nReceived:%s\r\n\r\n",str);

}
```

> **Примечание.**
>
> **Void _stdcall myfunc** вызывается в отдельном потоке (не в контексте основного потока приложения).

Разбор получаемых сообщений устанавливается разработчиком в соответствии с требованиями интеграции.

2.1.3.2 SendMsg

Для передачи сообщения ядру системы используется функция:

```
BOOL SendMsg (LPCTSTR id, LPCTSTR msg)
```

Описание параметров функции SendMsg приведено в таблице.

Параметр	Описание	Пример
LPCTSTR id	идентификатор подключения, указанный при вызове функции **Connect**	
LPCTSTR msg	текст сообщения	

```
CString port = "900";

CString ip = "127.0.0.1";

CString id = "2";

BOOL IsConnect = Connect(ip, port, id,
myfunc);

if (!IsConnect)

{

// не удалось подключиться

AfxMessageBox("Error");

Return;

}

SendMsg(id,"CAM|1|REC");  // поставить
камеру 1 на запись


Disconnect (id);
```

Если сообщение отправлено, функция возвращает TRUE, иначе – FALSE.

2.1.3.3 Disconnect

Каждое созданное соединение должно быть разорвано с помощью функции **Disconnect**:

```
void Disconnect (LPCTSTR id)
```

где **LPCTSTR id** – идентификатор подключения, указанный при вызове функции **Connect.**

Если разрыв соединения осуществляется со стороны ПК *Интеллект*, то Callback-функция принимает значение **DISCONNECTED**.

> **Примечание.**
>
> Пример использования функции **Disconnect** приведен в разделе SendMsg.

2.1.3.4 Другие функции

На странице:

Ниже перечислены дополнительные функции, объявленные в заголовочном файле iidk.h. Из них не рекомендуются к использованию функции Connect4, SendData4, SendFile, GetMsg. Они созданы для внутреннего пользования. Функция Connect2 не используется.

2.1.3.4.1 Connect3

```
BOOL Connect3(LPCTSTR ip, LPCTSTR port, LPCTSTR id, iidk_callback_func* lpfunc,
                              DWORD user_param,int async_connect,DWORD connect_attempts)
```

Параметр	Описание
ip	IP-адрес Сервера ПК *Интеллект*,
port	Порт TCP/IP, через которое происходит подключение
id	Идентификатор подключения slave, для видео
lpfunc	Callback-функция, принимающая сообщения от ПК *Интеллект*
user_param	Дополнительный параметр, который будет приходить в Callback-функцию, чтобы разделить слейвы, если функция одна на всех.
async_connect	0 - синхронный режим подключения, функция возвращает TRUE, если подключение выполнено -1 - асинхронный режим подключения, функция всегда возвращает FALSE, если подключение выполнено, то генерируется событие CONNECTED Любое другое значение - сначала используется синхронный режим, в случае неудачи асинхронный.
connect_attempts	Количество попыток подключения

2.1.3.4.2 SendReactToCore

Функция предназначена для отправки реакции в указанное ядро.

```
BOOL SendReactToCore(LPCTSTR id, LPCTSTR msg)
```

Параметр	Описание
id	Идентификатор подключения ядра

msg	Отправляемое сообщение. Формат сообщения аналогичен SendMsg.

2.1.3.4.3 IsConnected

IsConnected возвращает TRUE, если в данный момент клиент подключен к серверу.

```
BOOL IsConnected();
```

2.1.3.4.4 Connect4

```
BOOL Connect4(LPCTSTR ip, LPCTSTR port, LPCTSTR id,  iidk_callback_func* lpfunc,
                              iidk_frame_callback_func* lpframe_func, iidk_user_data_func*
iidk_user_data_func,
                              DWORD user_param,int async_connect,DWORD connect_attempts);
```

Параметр	Описание
ip	IP-адрес Сервера ПК *Интеллект*,
port	Порт TCP/IP, через которое происходит подключение
id	Идентификатор подключения ядра, для видео
lpfunc	Callback-функция, принимающая сообщения от ПК *Интеллект*
lpframe_func	Callback-функция, принимающая видеокадры
iidk_user_data_func	Callback-функция для данных, посылаемых при помощи функции SendData4
user_param	Дополнительный параметр, который будет приходить в Callback-функцию, чтобы разделить ядра, если Callback-функция одна на все ядра.
async_connect	0 - синхронный режим подключения, функция возвращает TRUE, если подключение выполнено -1 - асинхронный режим подключения, функция всегда возвращает FALSE. Если подключение выполнено, то генерируется событие CONNECTED Любое другое значение - сначала используется синхронный режим, в случае неудачи асинхронный режим.
connect_attempts	Количество попыток подключения

2.1.3.4.5 SendData4

Данная функция используется для посылки CUserNetObject, ее назначение - отправка "сырых данных".

```
BOOL SendData4(LPCTSTR id, int nIdent,BYTE *pBuffer,DWORD dwSize);
```

Параметр	Описание
id	Идентификатор подключения ядра
nIdent	Уникальный идентификатор данных

pBuffer	Пересылаемые данные
dwSize	Размер массива данных

2.1.3.4.6 SendFile

Функция служит для пересылки файла.

```
BOOL SendFile(LPCTSTR id, LPCTSTR file_from, LPCTSTR file_to)
```

Параметр	Описание
id	Идентификатор подключения ядра
file_from	Адрес, по которому находится файл для пересылки
file_to	Адрес, по которому следует записать файл.

2.1.3.4.7 GetMsg

Функция служит для выборки пришедших сообщений, которые помещаются в очередь, если Callback-функция не указана.

```
BOOL GetMsg(LPTSTR msg, DWORD& cb)
```

Параметр	Описание
msg	Получаемое сообщение
cb	Длина сообщения

2.1.4 Синтаксис отправляемых сообщений

2.1.4.1 Синтаксис сообщений

Сообщения, отправляемые ядру, должны иметь следующий вид:

CORE||DO_REACT|source_type<ТИП ОБЪЕКТА>,source_id<ИДЕНТИФИКАТОР ОБЪЕКТА>,action<ДЕЙСТВИЕ> [,params<КОЛ-ВО ПАРАМЕТРОВ>,param0_name<ИМЯ ПАРАМЕТРА_0>,param0_val<ЗНАЧЕНИЕ ПАРАМЕТРА_0>]

Ниже приведен синтаксис сообщения, содержащего 2 параметра.

CORE||DO_REACT|source_type<ТИП ОБЪЕКТА>,source_id<ИДЕНТИФИКАТОР ОБЪЕКТА >,action<ДЕЙСТВИЕ>,params<2>,param0_name<ИМЯ ПАРАМЕТРА_0>,param0_val< ЗНАЧЕНИЕ ПАРАМЕТРА _0>,param1_name<ИМЯ ПАРАМЕТРА_1>,param1_val<ЗНАЧЕНИЕ ПАРАМЕТРА_1>

Описание параметров сообщения приведено в таблице.

Параметр	Описание
source_type<obj>	тип объекта (см. DDI-файл, секцию [OBJTYPE])
source_id<id>	идентификационный номер объекта, заданный при создании объекта в ПК *Интеллект* (см. дерево настроек в ПК *Интеллект*)

Параметр	Описание
action<react>	действие (см. DDI-файл, секцию [REACT])
params<number>	число передаваемых параметров в десятичном формате
param0_name<str1>	имя параметра
param0_val<str2>	значение параметра

> **Примечание.**
>
> Для работы с DDI-файлами предпочтительно использовать программу ddi.exe (см. раздел Использование утилиты ddi.exe для работы с DDI-файлами).

Пример. Отправление сообщения с командой перевода телеметрии в предустановку 4.

```
CString msg=

"CORE||DO_REACT|
source_type<TELEMETRY>,source_id<1.1>,action<GO_PRESET>,params<2>,param0_name<preset>,param0_val<4>,param1_name<tel_prior>,param1_val<2>";

 SendMsg(id,msg);
```

2.1.4.2 Синтаксис сообщений (900 порт)

Сообщения, отправленные на 900 порт, передаются видеоподсистеме напрямую, поэтому сообщения имеют другой синтаксис.

Сообщения, отправляемые видеоподсистеме, имеют следующий вид:

ТИП ОБЪЕКТА|ИДЕНТИФИКАТОР ОБЪЕКТА|ДЕЙСТВИЕ [|ПАРАМЕТР<ЗНАЧЕНИЕ>]

Ниже описан синтаксис сообщения для видеоподсистемы, содержащего n-ое количество параметров.

ТИП ОБЪЕКТА|ИДЕНТИФИКАТОР ОБЪЕКТА|ДЕЙСТВИЕ [|ПАРАМЕТР 1<ЗНАЧЕНИЕ>,ПАРАМЕТР 2<ЗНАЧЕНИЕ>,…,ПАРАМЕТР N<ЗНАЧЕНИЕ>]

> **Внимание!**
>
> Через 900 порт можно управлять только объектами типа GRABBER, CAM и MONITOR.

Описание параметров сообщения представлено в таблице:

Параметр	Описание
Тип объекта	Тип объекта (GRABBER, CAM или MONITOR)
Идентификатор объекта	Идентификационный номер объекта, заданный при создании объекта в ПК *Интеллект*
Действие	Действие (команда)
Параметр <Значение>	Имя параметра. В треугольных скобках задается значение параметра

Пример 1. Постановка камеры 1 на запись.

```
CString msg = "CAM|1|REC";

SendMsg (id,msg);
```

Пример 2. Запись видео со всех видеокамер на локальный диск «C:».

```
CString msg = "GRABBER|1|SET_DRIVES|drives<C:\>" ;

SendMsg(id,msg);
```

Примечание

Для выполнения команды **SET_DRIVES** необходимо указать идентификационный номер любой устройства видеоввода, созданной в системе.

Примечание

Команда **SET_DRIVES** не меняет настройки записи видеосигнала в архив, заданные в системе.

2.1.4.3 Использование классов Event и React

Для работы с сообщениями можно использовать прилагаемые классы: *Event* и *React,* объявленные в файле msg.h.

Сообщение, составленное без использования классов	Сообщение, составленное с помощью класса React
CString msg = "CORE||DO_REACT|source_type<TELEMETRY>,source_id<1.1>, action<GO_PRESET>,params<2>,param0_name<preset>,param0_val<4>, param1_name<*tel_prior*>,param1_val<2>"; SendMsg(id,msg);	**React react("TELEMETRY","1.1","GO_PRESET"); react.SetParamInt("preset",4); react.SetParamInt("*tel_prior*",2); SendMsg(id,react.MsgToString().c_str());**

Примечание.

Файлы msg.h и msg.cpp содержатся в папке Misc.

2.1.5 Примеры управления объектами системы

Внимание!

Команды и параметры объектов подробно описаны в документе Руководство по программированию.

2.1.5.1 Добавление, изменение и удаление объектов системы

На странице:

Добавление, изменение и удаление объектов системы выполняется с помощью команд:

1. **CORE||CREATE_OBJECT** – для создания нового объекта.
2. **CORE||UPDATE_OBJECT** – для изменения существующего объекта или создания нового.
3. **CORE||DELETE_OBJECT** – для удаления объекта.

2.1.5.1.1 Добавление пользователя в отдел

Ниже приведено сообщение, в результате обработки которого в отдел будет добавлен пользователь с заданными параметрами:

CORE||CREATE_OBJECT|
objtype<PERSON>,objid<12341>,parent_id<1>,surname<Tim>,name<Kovac>,card<12362>,facility_code<0>

В ответ на эту команду IIDK вернет сообщение вида:

CORE||CREATE_OBJECT|card<1234>,objtype<PERSON>,guid_pk<{281A172C-62D2-EA11-A54B-B06EBF811A34}
>,facility_code<122>,surname<Tim>,module<iidk_client_test_x64.exe>,time<16:42:53>,parent_id<1>,fraction<797>,date<30-07-20>,name<Kovac>,owner<QA-T49>,slave_id<QA-T49.11>,objid<12341>

Это необходимо для того, чтобы получить идентификатор созданного объекта в параметре objid<>.

2.1.5.1.2 Добавление и удаление устройства видеоввода

Добавление объекта выполнятся с помощью команды **UPDATE_OBJECT,** если в системе отсутствует объект с указанными значениями для параметров **objtype** и **objid**.

CORE||UPDATE_OBJECT|objtype<GRABBER>,objid<12>,core_global<0>,parent_id<SLAVAXP>,name<Устройство видеоввода 1>,params<5>,param0_name<format>,param0_val<NTSC>,param1_name<mode>,param1_val<1>,param2_name<chan>,param2_val<2>,param3_name<type>,param3_val<FX 4>,param4_name<resolution>,param4_val<0>

Получив следующее сообщение, система изменит имя созданного объекта:

CORE||UPDATE_OBJECT|objtype<GRABBER>,objid<12>,core_global<0>,parent_id<SLAVAXP>,name<Устройство **2>**

Для удаления объекта и всех его дочерних объектов используется команда **DELETE_OBJECT**:

CORE||DELETE_OBJECT|objtype<GRABBER>,objid<12>

2.1.5.2 Особенности работы с системой в многопользовательском режиме

На удаленном компьютере должен быть установлен (тип установки – **Клиент**) и запущен ПК *Интеллект,* для того чтобы обмениваться сообщениями с Сервером.

Если в ПК *Интеллект* созданы пользователи и настроены права доступа, то передаваемые сообщения, требующие ответа от ядра системы, должны содержать параметр **receiver_id<ID>**, где ID – это идентификационный номер объекта **Интерфейс IIDK** в системе.

CORE||GET_CONFIG|objtype<CAM>,objid<1>,receiver_id<1>

// Возвращает параметры объекта «Камера 1»

Чтобы получить идентификатор пользователя и его прав по его логину и паролю, используется функция CHECK_USER. Примеры использования:
CORE||CHECK_USER|password<1>,login<1>
CORE||CHECK_USER|pass_key<1373503546>,login<1> (crc32 из БД)
CORE||CHECK_USER|md5<bf03b1605e3c83978514f2a6546eef50> (md5 из БД)
Ответ:
ACTIVEX|1|USER_RIGHTS|rights_id<1>,user_id<1>

В случае, если указан неправильный пароль, ответ приходит с задержкой в 1 секунду.

2.1.5.3 Определение компьютера, на котором был выгружен ПК Интеллект (через 1030 порт)

В случае выгрузки ПК *Интеллект* в Callback-функцию придет сообщение, где параметру **action** присвоено значение **DISCONNECTED**:

ACTIVEX|12|EVENT|SOCKET<>,MMF<>,objaction<DISCONNECTED>,TRANSPORT_TYPE<MMF>,core_global<1>, action<DISCONNECTED>, module<slave.exe>, objtype<SLAVE>,__slave_id<SLAVAXP.12>, objid<SLAVAXP>,owner<SLAVAXP>,TRANSPORT_ID<1111>,time<12:41:16>,date<23-09-02>

Данное сообщение содержит имя компьютера, на котором был выгружен ПК *Интеллект,* дату и время, когда это действие произошло.

2.1.5.4 Вывод видеокамеры на монитор

Система удалит все камеры с монитора и вызовет указанную видеокамеру, получив следующее сообщение:

CORE||DO_REACT|
source_type<MONITOR>,source_id<1>,action<REPLACE>,params<4>,param0_name<slave_id>,param0_val<SLAVA>,param1_name<cam>,param1_val<1>,param2_name<control>,param2_val<1>,param3_name<name>,param3_val<>

При подключении через 900 порт действие, описанное выше, выполняется с помощью сообщения:

MONITOR|1|REPLACE|slave_id<SLAVA>,cam<1>,control<1>

2.1.5.5 Получение параметров объекта (через 1030 порт). GET_CONFIG

Пример использования команды **GET_CONFIG** приведен ниже.

CORE||GET_CONFIG|objtype<CAM>,objid<1>

В возвращаемом сообщении будут содержаться все параметры указанного объекта:

ACTIVEX|12|OBJECT_CONFIG|
rec_priority<0>,mask0<>,decoder<0>,mask1<>,flags<>,mask2<>,compression<3>,sat_u<5>,mask3<>,proc_time<>,hot_rec_period<>,mask4<>,telemetry_id<>,manual<1>,region_id<1.1>,contrast<5>,md_mode<0>,md_size<5>,audio_type<>,pre_rec_time<0>,config_id<>,bright<7>,alarm_rec<0>,audio_id<>,rec_time<>,hot_rec_time<2>,activity<>,mux<0>,parent_id<1>,objtype<CAM>,type<>,__slave_id<SLAVAXP.12>,objid<1>,name<Камера 1>,objname<Камера 1>,color<1>,priority<0>,md_contrast<5>

> **Примечание.**
>
> Если убрать параметр **objid**, то в Callback-функцию вернется конфигурация всех объектов заданного типа.

Пример. Получить данные пользователя по его идентификатору.

CORE||GET_CONFIG|objtype<PERSON>,objid<1>

В ответ придет сообщение, среди параметров которого находится требуемая информация, такая как имя пользователя, номер карты доступа и прочее:

ACTIVEX|1|OBJECT_CONFIG|
pnet3_sound<0>,galaxy_dual_focus<0>,auto_pass_type<>,galaxy_pin_change<0>,external_id<>,card_date<26.05.2017 10:57:06>,galaxy_tag_link<0>,rubeg8_zone_id<>,levels_times<>,expired<>,hid_escort_id<>,objtype<PERSON>,level2_id<>,galaxy_group_choice<0>,
who_level<>,hid_use_extended_access<0>,visit_purpose<>,**card<1234>**,email<>,galaxy_timer_schedule<0>,galaxy_menu_option<0>,aiu_holiday<0>,
area_id<>,aiu_alarm<0>,objname<Пользователь 1>,surname<>,who_card<>,auto_brand<>,pnet3_alarm<0>,card_loss<0>,
facility_code<432>,galaxy_temp_code<0>,post<>,when_area_id_changed<>,drivers_licence<>,bolid_in_device<0>,pnet3_acs_counter<0>,
temp_levels_times<>,temp_card<>,pnet3_no_entry<0>,location<>,temp_level_id<>,patronymic<>,teleph_work<>,department<>,galaxy_keypad<0>,
_TRANSPORT_ID<>,finished_at<>,aiu_ksd_type<>,all_param<>,galaxy_template<0>,tabnum<>,parent_id<1>,pur<>,galaxy_duress<0>,pnet3_no_exit<0>,
galaxy_dual<0>,hid_pin_exempt<0>,pnet3_counter<0>,whence<>,schedule_id<>,hid_enable_pin_commands<0>,galaxy_dual_access<0>,
pnet3_block<0>,passport<>,person<>,galaxy_menu_choice<0>,flags<0>,auto_number<>,phone<>,pin<>,rubeg8_AccessToBCPTimeZoneNumber<>,
begin_temp_level<>,pnet3_master<0>,aiu_mark<0>,end_temp_level<>,visit_birthplace<>,galaxy_menu_level<>,visit_document<>,pnet3_guard<0>,
pnet3_black<0>,aiu_kso_type<>,is_apb<0>,**name<Пользователь 1>**,pnet3_temp<0>,started_at<>,level_id<>,_marker<>,
bolid_user_type<>,owner_person_id<>,card_type<>,is_active_temp_level<0>,begin<>,hid_line_tag<>,guid<{5B358685-E041-E711-BCC6-DA0AE28E0C17}>,
pnet3_guest<0>,is_locked<0>,objid<1>,marketing_info<>,comment<>,aiu_vpu_arm<0>,visit_reg<>,bolid_in_pku<0>,pnet3_2cards_mask<0>

> **Примечание.**
>
> Данные пользователя также можно получить напрямую через запрос к базе данных ПК *Интеллект* из таблицы OBJ_PERSON. В этом случае можно использовать для выбора требуемого пользователя также номер карты. Подробнее о работе с базой данных ПК *Интеллект* см. Руководство Администратора, раздел Приложение 4. Администрирование базы данных программного комплекса Интеллект.

2.1.5.6 Получение информации о состоянии объекта. GET_STATE и GET_LIST

Для получения информации о состоянии объекта используется команда **GET_STATE:**

CORE||GET_STATE|objtype<CAM>,objid<1>

В результате возвратится строка:

ACTIVEX|12|OBJECT_STATE|objtype<CAM>,__slave_id<SLAVAXP.12>,objid<1>,state<DISARM_DETACHED>

Состояние указанного объекта будет представлено значением параметра **state** – одно из состояний, указанных в DDI-файле для выбранного объекта.

При подключении через 900 порт запрос состояний объектов выполняется с использованием команды **GET_LIST:**

CAM||GET_LIST

> **Примечание.**
>
> Независимо от того, указан идентификационный номер объекта или нет, команда возвратит состояния всех объектов заданного типа.

Возвращаемые сообщения имеют вид:

CAM|1|SETUP|rec_priority<0>,is_armed<0>,is_recorded<0> , bt<0> , slave_id<SLAVAXP>, compression<3>,sat_u<5>, proc_time<0>, hot_rec_period<0> , manual<1>, telemetry_id<>, is_detached<1> , contrast<5>, md_size<5>,md_mode<0>, is_alarmed<0>, audio_type<>, pre_rec_time<0>, bright<7>, audio_id<>, rec_time<0>, alarm_rec<0>, hot_rec_time<2>, mux<0>, parent_id<1>, __slave_id<SLAVAXP>, priority<0>, mask<>, color<1>,md_contrast<5>, is_ring<1>, fs_error<0>

Состояния в сообщении представлены следующим образом: **is_state<val>,** где **state** – имя состояния объекта (см. DDI-файл); **val** – принимает значение 1, если объект находится в соответствующем состоянии, иначе – 0.

> **Примечание.**
>
> Параметр **is_ring<>** говорит о том, ведет ли камера запись в архив по кольцу. Параметр **fs_error** принимает значение 1 при наличии проблем с записью (например, не удается удалить старую папку при записи по кольцу).

2.1.5.7 Вывод информационного сообщения. SET_STATE

Для вывода информационного сообщения на дисплей главной панели управления ПК *Интеллект* используется команда SET_STATE:

CORE||SET_STATE|name<POS 1>,value<Can't open port COM4>

Результат обработки сообщения системой представлен на рисунке.

Отображение сообщения на дисплее главной панели управления ПК *Интеллект*:

Удаление информационного сообщения с дисплея выполняется следующим образом:

CORE||SET_STATE|name<POS 1>,value<>

2.1.5.8 Работа с живым и архивным видео

Для получения живого видео с Камеры 1 следует отправить на порт 900 сообщение:

CAM|1|START_VIDEO|compress<1>
Здесь compress<> – степень компрессии, от 0 до 5. В ответ на это сообщение начнут приходить кадры видео. Пример программной обработки поступающих кадров можно найти в демо-комплекте, доступном для скачивания на странице Руководство по интеграции аппаратно-программных модулей.

Для получения архивного видео с Камеры 1 следует отправить на порт 900 следующие сообщения:

CAM|1|ARCH_FRAME_TIME|time<dd-mm-yy HH:MM:SS.FFF> – для установки времени, начиная с которого требуется просматривать архив.

CAM|1|PLAY|compress<> – для получения архивного видео. Работа с архивным видео осуществляется таким же образом, как с живым.

Для того, чтобы получить список временных интервалов, содержащих видеозаписи за определенную дату, необходимо послать на порт 900 следующее сообщение:

CAM|id|ARCH_GET_INTERVALSREC|date<>

Параметр date<> может принимать значение date<dd-mm-yy|> или быть оставлен пустым. В первом случае будут запрошены интервалы за указанную дату, во втором – даты, за которые присутствует архив. В результате будет получено сообщение вида

Event: CAM|id|SET_INTERVALSREC|intervals<>,date<>,timezone<>

Значение параметра intervals<> имеет следующий вид: intervals<begin1 end1\nbegin2 end2…\nbeginN endN|date1\ndate2…\ndateN\n>

Время начала и время конца разделяется одним пробелом (код 0x20), интервалы отделяются друг от друга символом переноса строки '\n'(код 0x0A).

- begin1, begin2, ... beginN – времена начал интервалов в формате HH:MM:SS (возвращается, если запрошена точная дата).

- end1, end2, ... endN – времена концов интервалов в формате HH:MM:SS(возвращается, если запрошена точная дата).
- date1, date2, ... dateN – даты для которых присутствуют записи в архиве (возвращается, если поле date в запросе пусто или отсутствует).

Параметр date<dd-mm-yy|> – это дата, за которую запрашивались интервалы, или пустое значение (date<>), если запрашивались даты за весь период.

Параметр timezone<> определяет смещение времени клиента (IIDK) относительно времени Сервера в минутах. Примеры:

- Если Сервер во временной зоне -1 UTC, то в ответ на команду CAM|1|ARCH_GET_INTERVALSREC| будет получено событие с параметром timezone<60>
- Если Сервер во временной зоне +3 UTC, то в ответ на команду CAM|1|ARCH_GET_INTERVALSREC| будет получено событие с параметром timezone<-180>

2.1.5.9 Управление телеметрией

Управление телеметрией через IIDK осуществляется при помощи обычных реакций, описанных в Руководстве по программированию в разделе TELEMETRY, например:

CORE||DO_REACT|source_type<TELEMETRY>,source_id<1.1>,action<LEFT>,params<1>,param0_name<tel_prior>,param0_val<3> – сообщений на порт 1030 для поворота объектива камеры влево с высоким приоритетом.

TELEMETRY|1.1|LEFT|speed<2>,tel_prior<3> – реакция на порт 1030 для поворота объектива камеры влево с высоким приоритетом и средней скоростью.

2.1.5.10 Операции со слоем карты

Команда задания размера и положения значка объекта **Камера 1** на слое 1 выполняется одним из следующих способов:

1. Посылкой сообщения на порт 1030 **CORE||DO_REACT|source_type<MAPLAYER>,source_id<1>,action<CUSTOMIZE_OBJECT>,params<7>,param0_name<x>,param0_val<200>,param1_name<y>,param1_val<200>,param2_name<objtype>,param2_val<CAM>,param3_name<objid>,param3_val<1>,param4_name<a>,param4_val<90>,param5_name<w>,param5_val<70>,param6_name<h>,param6_val<80>**
   Здесь x, y, w, h – координаты и размер значка объекта на карте.
   a – угол наклона значка.
2. Посылкой реакции на порт 1030
   MAPLAYER|1|CUSTOMIZE_OBJECT|x<200>,y<200>,objtype<CAM>,objid<1>,a<90>,w<70>,h<80>

Вывод слоя 1 в окне интерактивной карты осуществляется одним из следующих способов:

1. Посылкой сообщения на порт 1030: **CORE||DO_REACT|source_type<MAPLAYER>,source_id<1>,action<ACTIVATE>**
2. Посылкой реакции на порт 1030: **MAPLAYER|1|ACTIVATE**-

2.1.5.11 Получение информации об очередях ядра командой GET_QUEUE_INFO

Для получения информации об очередях ядра используется команда GET_QUEUE_INFO:

CORE||GET_QUEUE_INFO

> **Примечание.**
>
> Если в системе несколько объектов **Интерфейс IIDK**, также может быть указан параметр receiver_id – см. Особенности работы с системой в многопользовательском режиме.

В ответ будет получена строка вида:

ACTIVEX|11|QUEUE_INFO|thread2<0>,thread1<0>,thread0<0>,posted_events<0>,_TRANSPORT_ID<>,server_reacts<0>, posted_reacts<0>,events_inwork<0>,coremanager_events<0>,thread3<0>

Параметры ответа соответствуют информации, отображаемой в окне **Статистика очередей** (открывается по нажатию Alt+F2). Описание параметров:

threadN<> – количество элементов в очереди потока N.

posted_events<> – количество входящих событий.

posted_reacts<> – количество реакций в работе.

coremanager_events<> – количество событий на отправку.

server_reacts<> – количество реакций на отправку.

events_inwork<> – количество событий в работе.

2.1.5.12 Проигрывание аудиоархива за период. Команда START_PLAY_TIME

Для проигрывания аудиоархива заданного микрофона за определенный период с помощью конкретного объекта **Динамик** следует выполнить команду вида

SPEAKER|{id}|START_PLAY_TIME|speaker_id<>,mic_id<>,time_start<>,time_end<>,cam_id<>

Примеры:

SPEAKER|1|START_PLAY_TIME|speaker_id<1>,mic_id<2>,time_start<28-02-20 14:23:24.092>,time_end<28-02-20 14:23:27.092>

SPEAKER|1|START_PLAY_TIME|speaker_id<1>,mic_id<2>,time_start<02-03-20 12:01:24.092>,time_end<02-03-20 12:04:27.092>,cam_id<1>

Если указан идентификатор камеры **cam_id<>**, то будет проигрываться аудиоархив, записанный синхронно с видеоархивом (т.е. из папки VIDEO). Если параметр cam_id<> не указан, то проигрывается архив из папки AUDIO.

speaker_id<> – идентификатор объекта Динамик

mic_id<> – идентификатор объекта Микрофон

time_start<> – время начала отрезка аудиоархива

time_end<> – время окончания отрезка аудиоархива

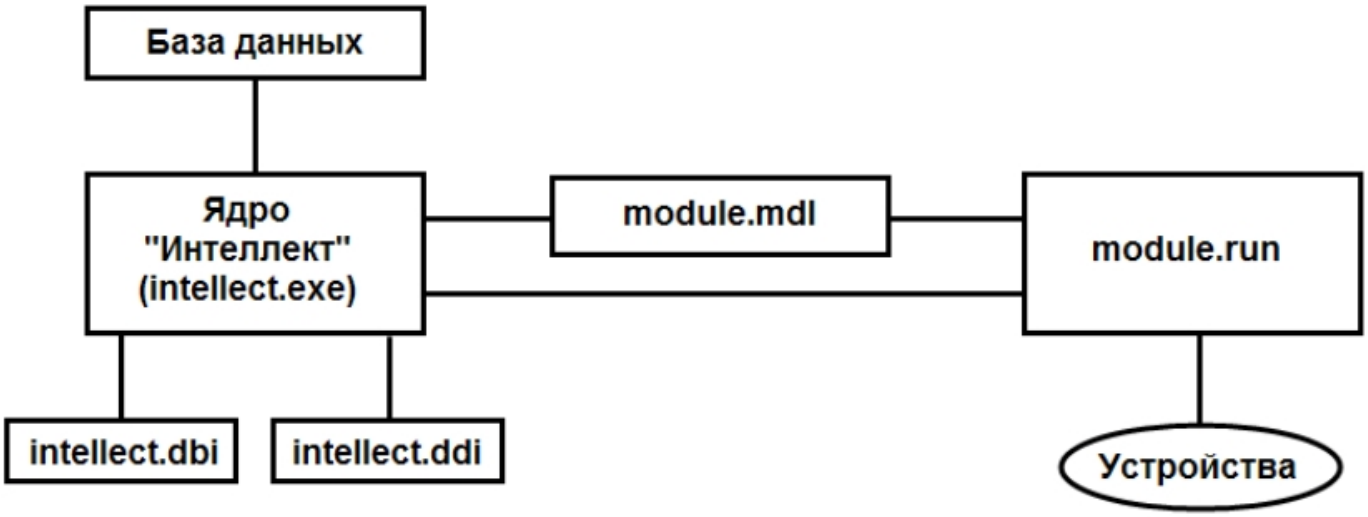
2.2 Интеграция аппаратно-программных модулей с ПК Интеллект

2.2.1 Общие сведения об интеграции аппаратно-программных модулей

Процесс интеграции аппаратно-программных (функциональных) модулей с ПК *Интеллект* состоит из следующих этапов:

1. Редактирование DBI-файла.
2. Редактирование DDI-файла.
3. Подготовка файла module.mdl, где module – имя интегрируемого модуля (данный файл является преобразованным DLL-файлом).
4. Подготовка исполнительного файла module.run, где module – имя интегрируемого модуля (этот файл является преобразованным exe-файлом).
5. Размещение module.mdl и module.run в каталоге *Интеллект*|*Modules*.

Схема взаимодействия функционального модуля с ядром системы представлена на рисунке.



DBI- и DDI-файлы содержат необходимую для функционирования ядра системы информацию об интегрированных функциональных модулях (объектах). DBI-файл содержит описание структуры конфигурационной базы данных ПК *Интеллект.* В DDI-файле хранится описание объектов и их параметров. При интеграции объекта в данные файлы заносят наименование, параметры интегрируемого объекта, связанные с ним системные события и реакции.

MDL-файлы обеспечивает работу с объектами одного типа: создание, изменение, удаление, изменение при настройке или в процессе работы параметров объекта и сохранение их в базе данных, выполнение некоторых специализированных операций с объектом. Также MDL-файл обеспечивает пересылку параметров созданных или измененных объектов исполнительному модулю (RUN-файлу) и хранит конфигурации настроечных панелей объектов.

Исполняемый RUN-файл осуществляет взаимодействие с устройствами, транслирует в ядро информацию о событиях, обеспечивает выполнение управления устройствами.

Далее описываются этапы интеграции модулей на примере демонстрационного модуля DEMO, эмулирующего работу с виртуальным оборудованием. Данный модуль включает в себя устройства с уникальными адресами для обращения к этим устройствам и их опроса. Таким образом, в системе будет существовать конфигурация, состоящая из 2 основных объектов: родительского объекта **DEMO** с параметром **COM-port** и дочернего объекта **DEMO_DEVICE** с параметром **Address**. В системе возможно выполнение определенного набора действий с устройствами и передача всех происходящих в них событий ядру системы.

2.2.2 Редактирование DBI-файла

Файл intellect.dbi содержит основной перечень таблиц и полей базы данных. Рекомендуется создавать собственный шаблон базы данных в отдельном файле – intellect.xxx.dbi, где xxx – уникальная часть имени файла. Использование отдельного файла позволяет избежать повторного включения таблиц и полей в случае обновления ПК *Интеллект*. При запуске программного комплекса DBI-файлы объединяются.

2.2.2.1 Добавление объектов в intellect.dbi

Добавление объектов в intellect.dbi выполняется следующим образом:

1. Открыть в текстовом редакторе файл intellect.dbi, расположенный в корневом каталоге ПК *Интеллект.*
2. Добавить в intellect.dbi объекты. Для этого необходимо в квадратных скобках указать имя, используемое для идентификации объекта, и далее объявить его поля. Синтаксис объявления полей имеет вид:

   <Имя поля>, <Тип> [, <Размер>]

Примечание.

Размер задается только полям с типом **CHAR.**

В таблице приведены обязательные поля для всех объектов ПК *Интеллект.*

Поле	Описание
id	Уникальный идентификатор объекта
name	Имя объекта
parent_id	Идентификатор родительского объекта
flags	Параметр для внутренних нужд системы

Внимание!

Поле **flags** не может использоваться внешними приложениями.

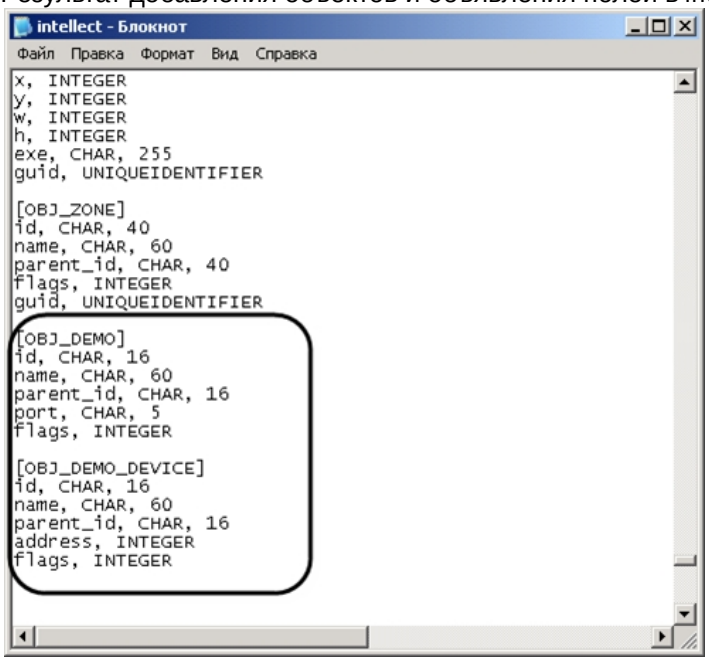
Допустимые типы данных описаны в таблице.

Тип данных	Описание
BIT	Используется для создания поля-флажка, принимающего логические значения «Да» или «Нет»
CHAR	Используется для полей, заполняемых небольшим количеством символов
DATETIME	Используется для полей, в которые вводятся дата и время. Маска для даты – гггг-мм-дд, для времени – чч:мм:сс.xxx
DOUBLE	Используется для полей, содержащих числа с плавающей запятой
INTEGER	Используется для полей, содержащих целые числа
TEXT	Используется для полей, содержащих текстовые строки

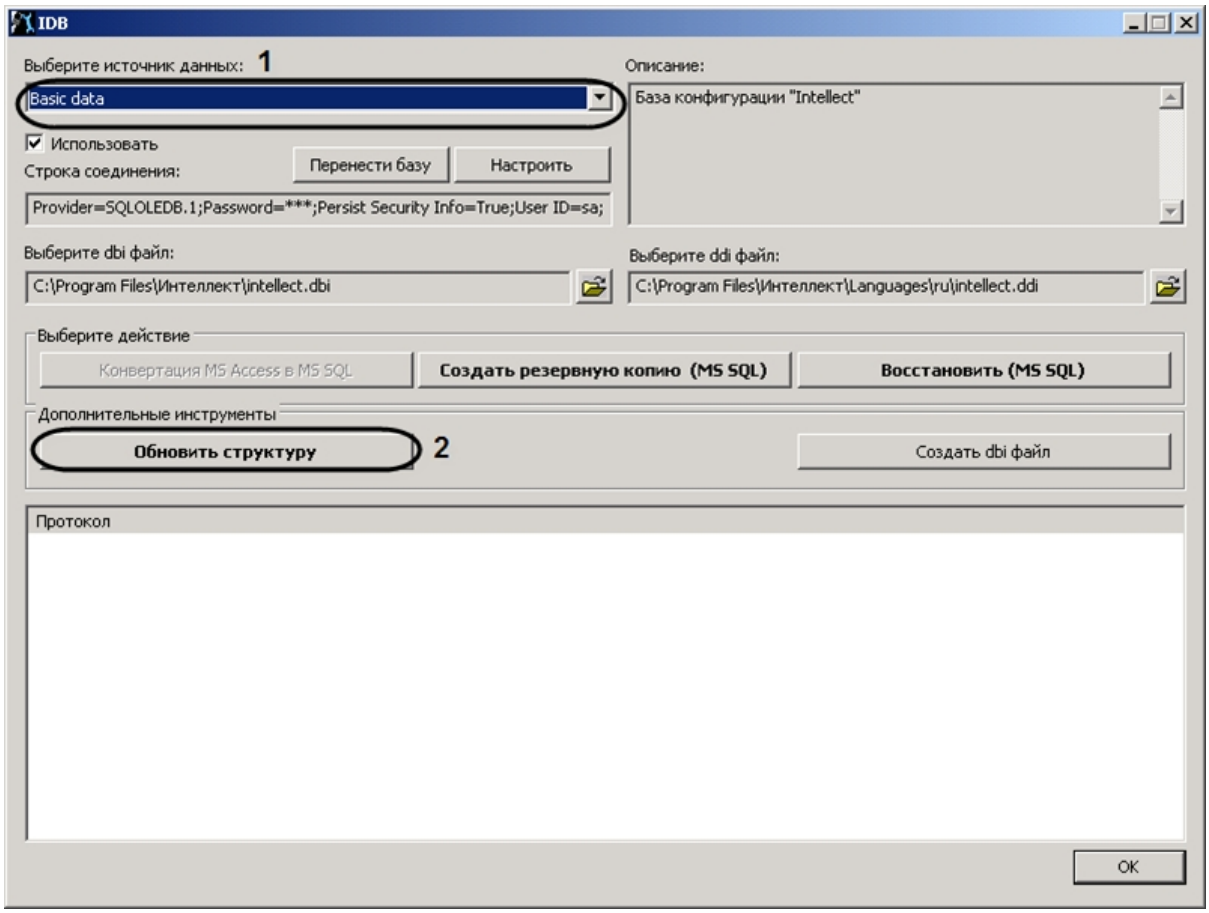
Для объектов демонстрационного модуля *DEMO*, кроме обязательных полей, добавлены поля:

a. **port** – адрес COM-порта;
b. **address** – адрес устройства.

Результат добавления объектов и объявления полей в intellect.dbi представлен на рисунке.

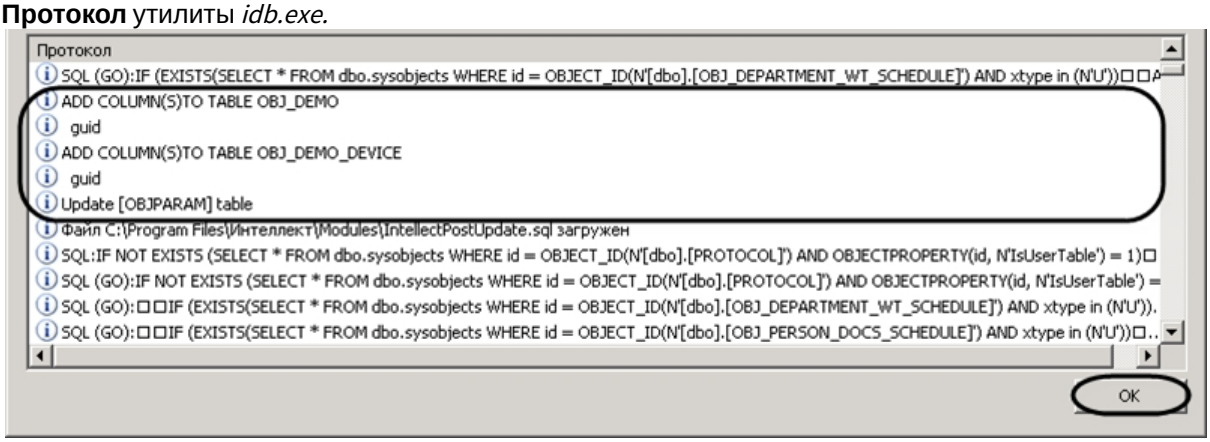


3. Сохранить изменения в файле intellect.dbi.
4. Запустить утилиту *idb.exe*, расположенную в корневом каталоге ПК *Интеллект.*



5. Из списка **Выберите источник данных:** выбрать **Basic data** (**1**).
6. Нажать кнопку **Обновить структуру** (**2**).
   Будет запущен процесс обновления структуры базы данных. Выполнение процесса отображается в окне

Протокол утилиты *idb.exe.*



7. Нажать кнопку **OK** для завершения работы с утилитой *idb.exe*.

В результате обновления структуры будут созданы таблицы в базе конфигурации *Intellect*.

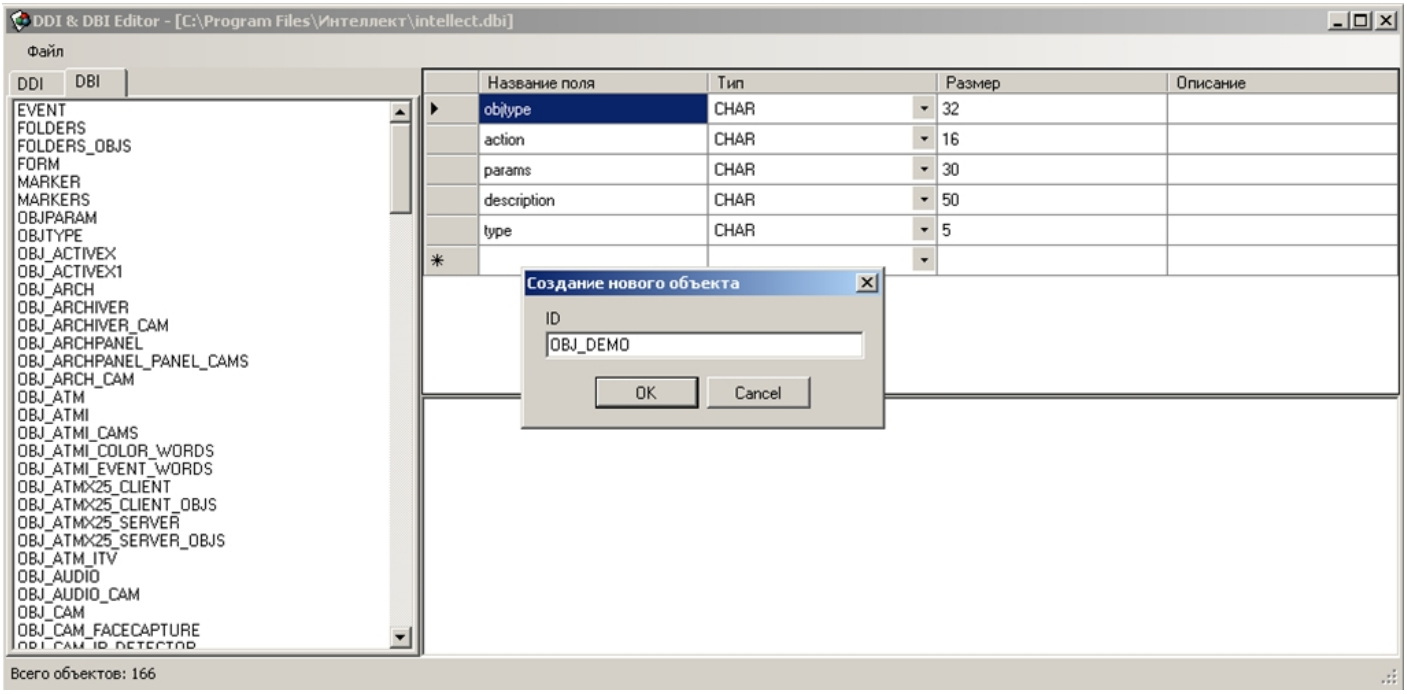
2.2.2.2 Использование утилиты ddi.exe для работы с DBI-файлами

Для добавления объекта в DBI-файл с помощью утилиты *ddi.exe* необходимо выполнить следующие действия:

1. Запустить утилиту *ddi.exe,* расположенную в каталоге *Интеллект|Tools.*
2. В окне утилиты перейти на вкладку **DBI**.
3. В меню **Файл** выбрать пункт **Открыть**. В результате выполнения операции появится диалоговое окно **Открыть.**
4. Выбрать файл intellect.dbi, расположенный в корне директории установки ПК *Интеллект.* В утилите *ddi.exe* отобразится список объектов.
5. В контекстном меню списка объектов выбрать пункт **Добавить** для добавления нового объекта.

> **Примечание.**
>
> Также возможно добавить объект с помощью клавиши **Insert.**

6. В открывшемся диалоговом окне ввести имя, используемое для идентификации объекта, в поле **ID** и нажать **OK**.



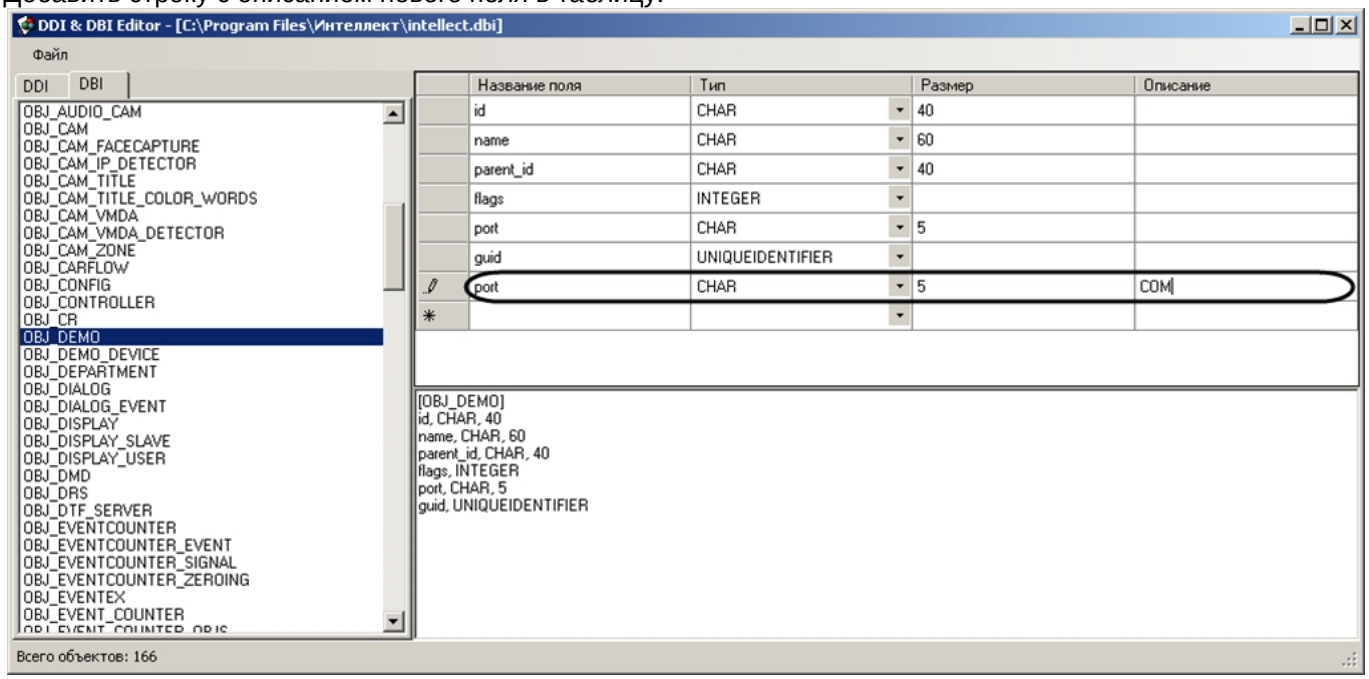
Примечание.

Созданному объекту будут автоматически добавлены обязательные поля (см. раздел Добавление объектов в intellect.dbi).

Добавление объекта в DBI-файл завершено.

Добавление поля выполняется следующим образом:

1. Выбрать созданный объект в левой части окна утилиты ddi.exe
2. Добавить строку с описанием нового поля в таблицу.



3. Сохранить внесенные изменения, выбрав в меню **Файл** пункт **Сохранить**.

Добавление поля выполнено.

Внимание!

После изменения DBI-файлов требуется обновить структуру базы данных с помощью утилиты idb.exe (см. раздел Добавление объектов в intellect.dbi).

2.2.3 Редактирование DDI-файла

DDI-файл представляет собой файл в формате xml, который содержит следующую информацию об объектах:

1. Реакции – действия, которые могут выполнять объекты.
2. События, которые могут генерировать объекты.
3. Состояния, в которых могут находиться объекты.
4. Правила перехода объектов из одного состояния в другое по определенным событиям.
5. Имена bmp-файлов, используемых для отображения объектов на *Карте*.

Файл intellect.ddi содержит свойства основных объектов ПК *Интеллект*. Для собственных объектов рекомендуется создавать отдельный файл – intellect.xxx.ddi, где xxx – уникальная часть имени файла. Использование отдельного файла позволяет избежать повторного включения свойств объектов в случае обновления ПК *Интеллект*. При запуске программного комплекса DDI-файлы объединяются.

В случае, если объект дублируется в нескольких ddi-файлах, при запуске ПК *Интеллект* применяются свойства объекта из последнего файла в соответствии с сортировкой файлов по имени. Например, если какой-либо объект

описан в файлах intellect.xxx.ddi, intellect.xxx1.ddi и intellect.xxx2.ddi, будут применены свойства из файла intellect.xxx2.ddi.

2.2.3.1 Добавление в intellect.ddi информации об объекте

В данном разделе приведен пример добавления в intellect.ddi информации об объекте **DEMO** с использованием текстового редактора.

Для добавления информации об объекте **DEMO** в intellect.ddi необходимо выполнить следующие действия:

1. Открыть в текстовом редакторе файл intellect.ddi, расположенный в каталоге *Интеллект|Languages|ru.*
2. В раздел **<DataSetDDI>** добавить дочерний элемент **<Objects>**, содержащий описание объекта.

```
<Objects>
  <ObjectName>DEMO</ObjectName>
  <VisibleName>Демо</VisibleName>
  <Events>
    <EventName>LOST</EventName>
    <EventDescription>Потеря связи</EventDescription>
    <IsSoundEnabled>false</IsSoundEnabled>
    <IsNetworkDisabled>false</IsNetworkDisabled>
    <IsProtocolDisabled>false</IsProtocolDisabled>
    <IsWindowsLogEnabled>false</IsWindowsLogEnabled>
  </Events>
  <Events>
    <EventName>RESTORE</EventName>
    <EventDescription>Восстановление связи</EventDescription>
    <IsSoundEnabled>false</IsSoundEnabled>
    <IsNetworkDisabled>false</IsNetworkDisabled>
    <IsProtocolDisabled>false</IsProtocolDisabled>
    <IsWindowsLogEnabled>false</IsWindowsLogEnabled>
  </Events>
  <Icons>
    <FileName>demo</FileName>
    <IconName>demo</IconName>
  </Icons>
  <States>
    <StateName>DETACHED</StateName>
    <ImgName>detached</ImgName>
    <IsStateFlashing>false</IsStateFlashing>
  </States>
  <States>
    <StateName>NORMAL</StateName>
    <ImgName>normal</ImgName>
    <IsStateFlashing>false</IsStateFlashing>
  </States>
  <Rules>
    <EventName>RESTORE</EventName>
    <FromStateName>DETACHED</FromStateName>
    <ToStateName>NORMAL</ToStateName>
  </Rules>
  <Rules>
    <EventName>LOST</EventName>
    <FromStateName>NORMAL</FromStateName>
    <ToStateName>DETACHED</ToStateName>
  </Rules>
</Objects>
```

Примечание

Для объекта **DEMO** отсутствует раздел **<Reacts>**, так как данный объект не выполняет никаких действий.

> **Примечание**
>
> Элементы DDI-файла подробно описаны в разделе ПРИЛОЖЕНИЕ 1. Описание структуры ddi-файла

3. Сохранить изменения в файле intellect.ddi.

Внесение в intellect.ddi информации об объекте **DEMO** завершено.

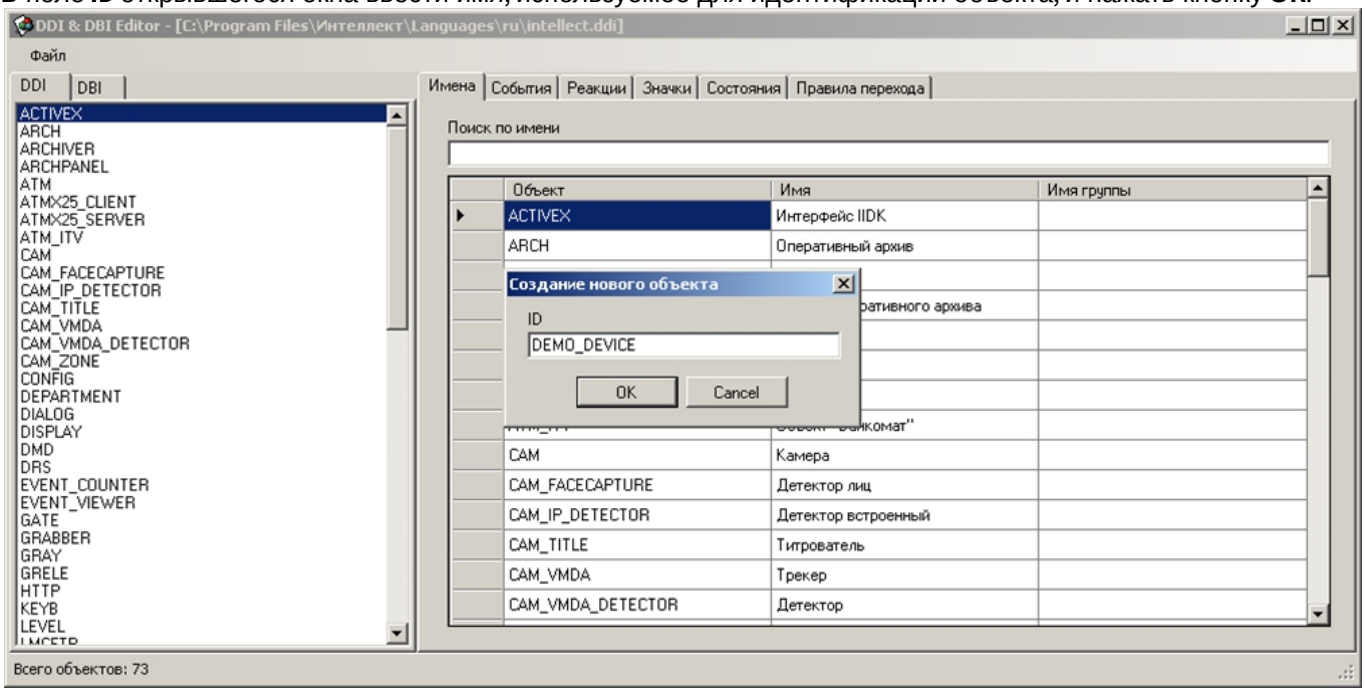
> **Внимание!**
>
> После изменения DDI-файлов требуется обновить структуру базы данных с помощью утилиты idb.exe (см. шаги 4-7 раздела Добавление объектов в intellect.dbi).

2.2.3.2 Использование утилиты ddi.exe для работы с DDI-файлами

В данном разделе приведен пример добавления в intellect.ddi информации об объекте **DEMO_DEVICE** с использованием утилиты *ddi.exe.*

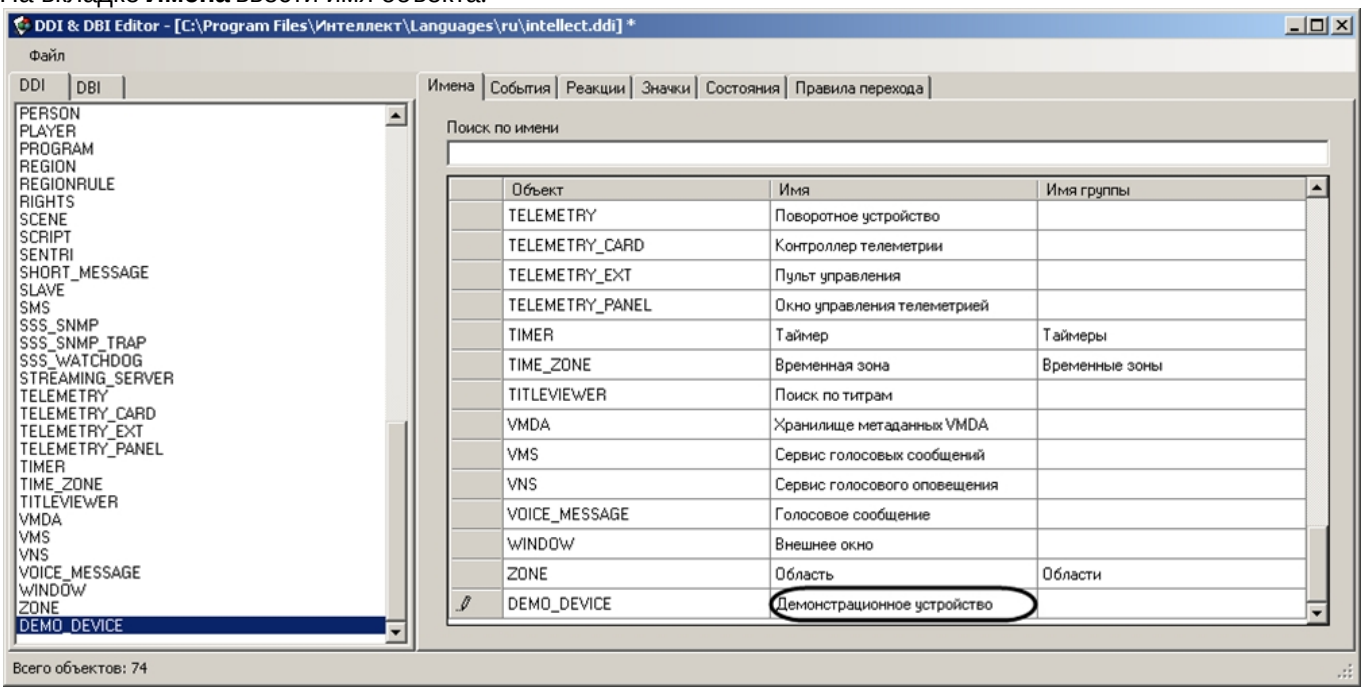
Для добавления в intellect.ddi информации об объекте **DEMO_DEVICE** необходимо выполнить следующие действия:

1. Открыть файл intellect.ddi в утилите одним мз следующих способов:
   a. Из утилиты ddi.exe:
      i. Запустить утилиту *ddi.exe,* расположенную в каталоге *Интеллект|Tools.*
      ii. В окне утилиты перейти на вкладку **DDI**.
      iii. В меню **Файл** выбрать пункт **Открыть**. В результате выполнения операции появится диалоговое окно **Открыть.**
      iv. Выбрать файл intellect.ddi, расположенный в каталоге *Интеллект|Languages|ru.* В утилите *ddi.exe* отобразится список объектов.
   b. Открыть каталог *Интеллект|Languages|ru* в проводнике Windows или любой другой программе для управления файлами и дважды щелкнуть по файлу intellect.ddi.
2. Добавить объект, выбрав в контекстном меню списка объектов пункт **Добавить** или нажав кнопку **Insert.**
3. В поле **ID** открывшегося окна ввести имя, используемое для идентификации объекта, и нажать кнопку **OK**.

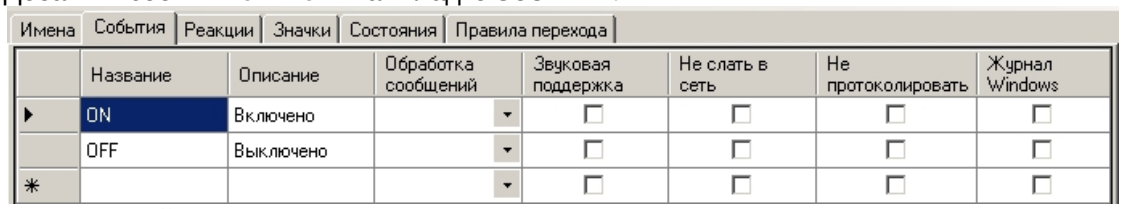


В результате выполнения операции объект **DEMO_DEVICE** отобразится в списке объектов.

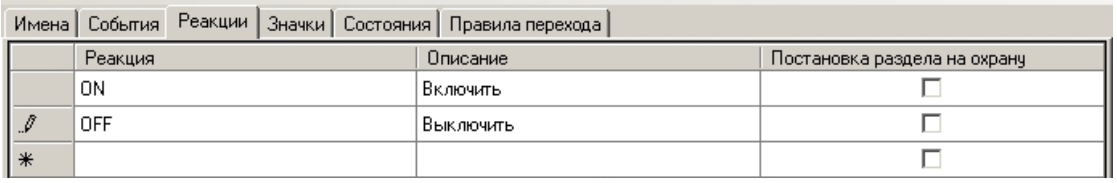
4. На вкладке **Имена** ввести имя объекта.



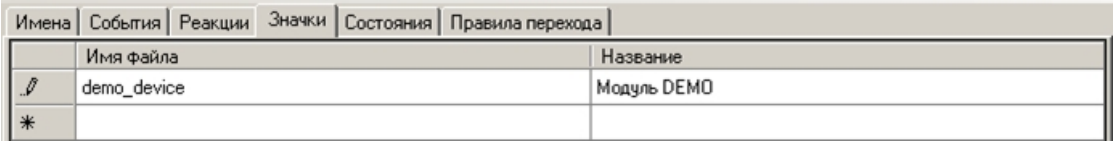
5. Добавить информацию об объекте **DEMO_DEVICE** на соответствующих вкладках.
   a. Добавить события **ON** и **OFF** на вкладке **События**.



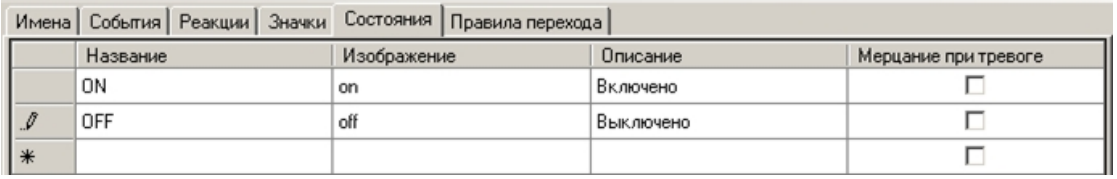
   b. Добавить действия **ON** и **OFF** на вкладке **Реакции**.



   c. На вкладке **Значки** указать часть имени bmp-файла, которая является идентификатором изображения. Идентификатор изображения позволяет использовать несколько bmp-файлов для представления на *Карте* объектов одного типа.



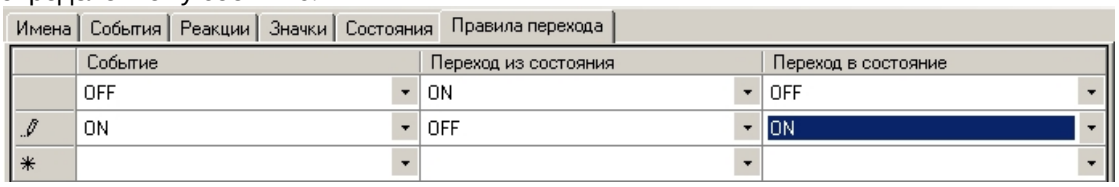
   d. На вкладке **Состояния** добавить состояния **ON** и **OFF**. Для отображения состояния объекта на *Карте* необходимо указать часть имени, которая является идентификатором состояния, соответствующего bmp-файла.



> **Примечание.**
>
> Каталог Интеллект\Bmp должен содержать bmp-файлы, имена которых составлены следующим образом:
> **<Идентификатор изображения>_<Идентификатор состояния>**
> Если идентификатор изображения не задан, то имя bmp-файла должно иметь вид:
> **<Идентификатор объекта>_<Идентификатор состояния>**

> **Примечание**
>
> Объекты на Карте могут быть отображены с помощью линий, т.е. без использования bmp-файлов. В этом случае, если изменяется состояние объекта, меняется цвет линии. Цвет (RGB) состоянию задается следующим образом:
> **<Состояние>$R:G:B**

e. На вкладке **Правила перехода** задать правило перехода из одного состояния в другое по определенному событию.



> **Примечание.**
>
> Если поле **Переход из состояния** оставить пустым, то переход будет осуществляться из любого состояния.

6. Сохранить изменения, выбрав в меню **Файл** пункт **Сохранить**.

Информация об объекте **DEMO_DEVICE** внесена.

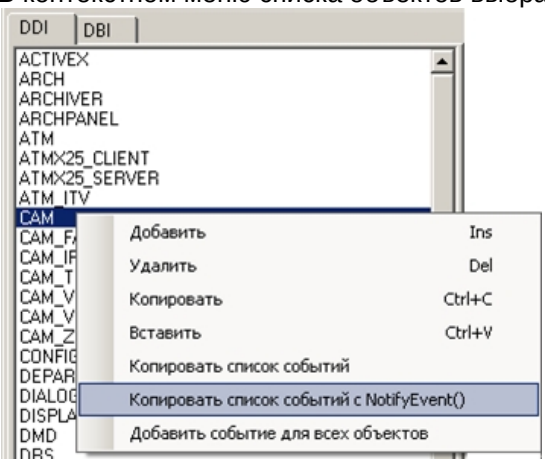
> **Примечание.**
>
> Поля таблиц утилиты ddi.exe подробно описаны в ПРИЛОЖЕНИЕ 1. Описание структуры ddi-файла

> **Внимание!**
>
> После изменения DDI-файлов требуется обновить структуру базы данных с помощью утилиты idb.exe (см. шаги 4-7 раздела Добавление объектов в intellect.dbi).

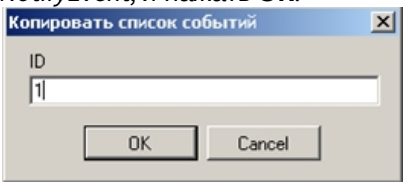
2.2.4 Дополнительные возможности утилиты ddi.exe

Утилита *ddi.exe* представляет собой удобный инструмент для удаления, добавления, редактирования и копирования в буфер обмена свойств объекта (событий, реакций и т.д). Дополнительно утилита позволяет копировать в буфер обмена события объекта в виде параметра функции *NotifyEvent*. Для этого необходимо выполнить следующие действия:

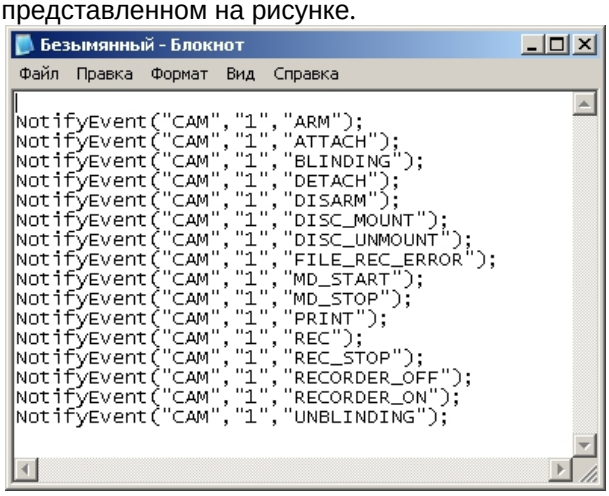
1. В контекстном меню списка объектов выбрать пункт **Копировать список событий с NotifyEvent()** .



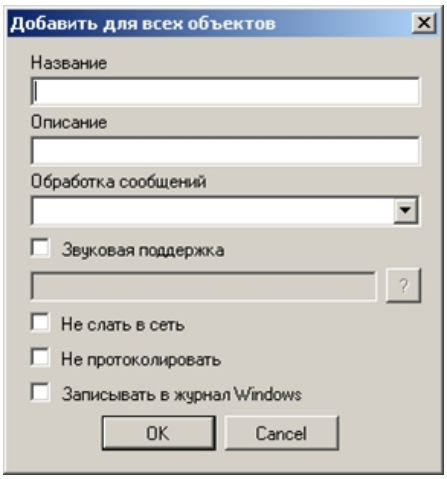
2. В открывшемся окне ввести идентификационный номер объекта, который следует использовать в функции *NotifyEvent*, и нажать **ОК**.



Копирование списка событий завершено. Буфер обмена будет содержать события объекта в виде, представленном на рисунке.



В случае, если требуется добавить событие всем объектам, следует в контекстном меню выбрать пункт **Добавить событие для всех объектов**. В результате выполнения операции будет открыто диалоговое окно **Добавить для всех объектов**, в котором параметрам создаваемого события задаются значения.



Для добавления объектов из других DBI- и DDI-файлов следует в меню **Файл** выбрать пункт **Вставить из файла**.

2.2.5 Разработка MDL-файла

Для создания mdl-файла необходимо использовать два класса:

1. *NissObjectDLLExt*. Все объекты наследуются от этого класса с переопределением его виртуальных методов.
2. *CoreInterface*. Методы класса используются для получения параметров объектов системы.

Объявленные классы и методы содержатся в заголовочном файле nissdlle.h. Код, содержащийся в файле nissdlle.h, представлен в разделе ПРИЛОЖЕНИЕ 2. Объявление классов NissObjectDLLExt и CoreInterface.

> **Примечание.**
>
> Под методами класса подразумеваются процедуры и функции, объявленные в теле класса.

Описание методов класса *NissObjectDLLExt* приведено в таблице.

Метод	Описание	Пример
CoreInterface* m_pCore	Указатель на интерфейс ядра	
virtual BOOL IsWantAllEvents ()	Возвращает TRUE, если необходимо в функции OnEvent получать события от всех объектов, FALSE - если только от своего объекта.	если указать “CAM,GRABBER”, то при изменении настроек этих объектов для объекта DEMO придут сообщения: DEMO|1|UPDATE_CAM|параметры камеры DEMO|1|UPDATE_GRABBER|параметры устройства видеоввода
virtual CString DescribeSubscribeObjectsList()	Через запятую указываются типы объектов, при изменении которых происходит уведомление текущего объекта	
virtual CString GetObjectType()	Возвращает тип объекта	`virtual CString GetObjectType()` `{ return "DEMO"; }`

Метод	Описание	Пример
virtual CString GetParentType( )	Возвращает тип родительского объекта	`virtual CString GetParentType()` `{ return "SLAVE"; }`
virtual int GetPos()	Возвращает позицию объекта в ключевом файле "*intellect.sec*". **Внимание! Этот параметр должен быть согласован с компанией «Ай Ти Ви групп»**	`virtual int GetPos()` `{ return -1; }` *Примечание. При запуске ПК Интеллект в демо-режиме функция возвращает **-1***
virtual CString GetPort()	Возвращает номер порта, через который будет происходить соединение и обмен сообщениями между объектом и ядром **Внимание! Этот параметр должен быть согласован с компанией «Ай Ти Ви групп»**	`virtual CString GetPort()` `{ return "1100"; }`
virtual CString GetProcessNam e()	Возвращает имя процесса. Используется ядром для поиска и автоматического запуска исполнительного модуля при старте системы и инициализации модуля	`virtual CString GetProcessName()` `{ return "demo"; }`
virtual CString GetDeviceType()	Определяет тип объекта и характер его поведения. ACD – объект этого типа получает все события, связанные с созданием, изменением и удалением следующих объектов: **Пользователи**, **Временная зона** и **Уровни доступа** ACD2– тип аналогичный ACD с дополнительной (обеспеченной ядром) возможностью автоматически удалять временные (с ограниченным сроком действия) карточки Тип ACR указывает на то, что объект является считывателем	Все объекты типа ACR доступны в раскрывающемся списке **Точка прохода**
virtual BOOL HasChild()	Возвращает TRUE, если у объекта есть дочерние объекты, иначе – FALSE	`virtual BOOL HasChild()` `{ return TRUE; }`
virtual UINT HasSetupPanel( )	Если у объекта имеется панель настроек, метод возвращает TRUE, иначе – FALSE	`virtual UINT HasSetupPanel()` `{ return TRUE; }`

Метод	Описание	Пример
virtual void OnPanelInit(CWnd*)	Используется при инициализации панели настроек объекта. В качестве параметра передается указатель на окно панели настроек	
virtual void OnPanelLoad(CWnd*,Msg&)	Используется при загрузке панели настроек для получения параметров объекта. В качестве параметра метода передается указатель на окно панели настроек и ссылка на сообщение, через которое осуществляется передача параметров объекта, и заполнение ими необходимых полей в панели настроек	`virtual void OnPanelLoad(CWnd* pwnd,Msg& params)` `{` `CString s;` `s = arams.GetParam("port");` `pwnd->GetDlgItem(IDC_PORT)->SetWindowText(s);` `}`
virtual void OnPanelSave(CWnd*,Msg&)	Используется при выгрузке панели настроек для сохранения параметров объекта. В качестве параметра метода передается указатель на окно панели настроек и ссылка на сообщение, через которое осуществляется передача параметров объекта и сохранение их в БД	`virtual void OnPanelSave(CWnd* pwnd,Msg& params)` `{` `CString s;` `pwnd-> GetDlgItem(IDC_PORT)-> GetWindowText(s);` `params.SetParam("port",s );` `}`
virtual void OnPanelExit(CWnd*)	Используется при закрытии панели настроек объекта. В качестве параметра передается указатель на окно панели настроек	

Метод	Описание	Пример
virtual void OnPanelButton Pressed(CWnd*, UINT)	Предназначен для обработки нажатий кнопок на панели настроек объекта. В качестве параметра передается указатель на окно панели настроек и идентификатор кнопки. *Примечание. Числовые значения идентификаторов кнопок должны начинаться с номера **1151**. Например, для кнопки **Test** идентификатор в файле Resource.h определяется как:* ***#define IDC_TEST 1151***	`Virtual void OnPanelButtonPressed (CWnd* pwnd,UINT id)` `{` `if(id==IDC_TEST)` `{ React react("DEMO",Id,"TEST");` `m_pCore->DoReact(react); }` `}`
	Если необходимо по нажатию кнопки вывести собственное диалоговое окно, созданное в этом же MDL-файле, то необходимо предварительно использовать код, указанный в примере	`HINSTANCE prev_hinst = AfxGetResourceHandle();` `HMODULE hRes = GetModuleHandle("demo.mdl");` `If (hRes) AfxSetResourceHandle (hRes);` `//Код вывода диалогового окна:` `CXXXDialog dlg;` `dlg.DoModal();` `AfxSetResourceHandle(prev_hinst);`
virtual BOOL IsRegionObject( )	Указывает на то, что объект будет поддерживать разделы ПК *Интеллект*. Разделы применяются для группировки объектов. Также они могут использоваться в системе отчетов	
virtual BOOL IsProcessObject ()	Указывает на то, что объект будет поддерживать возможность одновременного запуска и параллельной работы нескольких исполнительных модулей. Например, это может использоваться для запуска отдельного модуля непосредственно для каждого COM-порта. *Примечание. Рекомендуется использовать один рабочий модуль. Это упростит отладку и модификацию модуля*	

Метод	Описание	Пример
virtual void OnCreate(Msg&)	Используется при создании объекта. В качестве параметра передается ссылка на сообщение, содержащее информацию по объекту. Здесь же указываются параметры по умолчанию	`virtual void OnCreate (Msg& msg)` `{` `msg.SetParam ("port","COM1");` `}`
virtual void OnInit(Msg&)	Используется при инициализации объекта. В качестве параметра передается ссылка на сообщение, содержащее информацию по объекту	`virtual void OnInit (Msg& msg)` `{` `OnChange (msg, msg);` `}`
virtual void OnChange(Msg &,Msg&)	Используется при изменении объекта. В качестве параметра передаются ссылки на сообщения, содержащие информацию по объекту до и после изменения соответственно	`virtual void OnChange(Msg& msg, Msg& prev)` `{` `React react (msg.GetSourceType(), msg.GetSourceId(),"INIT");` `react.SetParam("port",msg.GetParam("port"));` `m_pCore->DoReact(react);` `}`
virtual void OnDelete(Msg&)	Используется при удалении объекта. В качестве параметра передается ссылка на сообщение, содержащее информацию по объекту	`virtual void OnDelete (Msg& msg)` `{` `React react (msg.GetSourceType(),`

Метод	Описание	Пример
		msg.GetSourceId(),"EXIT"); m_pCore->DoReact(react); }
virtual void OnEnable(Msg&)	Предназначен для обработки нажатия кнопки **Disable** на панели ПК *Интеллект* при включении объекта. В качестве параметра передается ссылка на сообщение, содержащая информацию по объекту	
virtual void OnDisable(Msg&)	Предназначен для обработки нажатия кнопки **Disable** на панели ПК *Интеллект* при выключении объекта. В качестве параметра передается ссылка на сообщение, содержащая информацию по объекту	

Метод	Описание	Пример
virtual BOOL OnEvent(Event&)	Служит для обработки событий, передаваемых в качестве параметра	virtual BOOL OnEvent(Event& event) { If (event.GetAction() == "ACCESS_IN" || event.GetAction() == "ACCESS_OUT") { Msg per = m_pCore-> FindPersonInfoByCard(event.GetParam("facility_code"), event.GetParam("card")); event.SetParam ("param0", ! per.GetSourceId().IsEmpty() ? per.GetParam("name") : event.GetParam("facility_code") + event.GetParam("card")); event.SetParam("param1", per.GetSourceId()); } Else If (event.GetAction() == "NOACCESS_CARD") { event.SetParam

Метод	Описание	Пример
		("param0",event.GetParam ("facility_code") + event.GetParam("card")); } return TRUE; }
virtual BOOL OnReact(React&)	Служит для обработки реакций, передаваемых в качестве параметра	

В глобальной функции *CreateNissObject(CoreInterface* core)* необходимо создать экземпляры описанных объектов, поместить их в массив *CNissObjectDLLExtArray* и возвратить указатель на объект этого массива. Через эту функцию необходимо получить указатель на интерфейс ядра, который в дальнейшем используется в объектах для обращения к методам данного интерфейса:

```
CNissObjectDLLExtArray* APIENTRY CreateNissObject(CoreInterface* core)

{

                              CNissObjectDLLExtArray* ar = new CNissObjectDLLExtArray;



                ar->Add(new NissObjectDemo(core));

                ar->Add(new NissObjectDemoDevice(core));



                              return ar;

}
```

После загрузки DLL-файла ядро вызывает функцию *CreateNissObject* и получает указатели на все используемые объекты.

Все панели настроек объектов хранятся в ресурсах в виде диалогов. Идентификаторы диалогов строятся по схеме **IDD_object_SETUP**, где **object** – это имя соответствующего объекта. Например, для объекта **DEMO** – это **IDD_DEMO_SETUP**, а для объекта **DEMO_DEVICE** – это **IDD_DEMO_DEVICE _SETUP.**

> **Примечание.**
>
> Для того чтобы в дереве настроек у объекта отображался свой собственный значок, необходимо в ресурсах DLL-файла создать **BITMAP** размером 14x14 с именем объекта.

2.2.5.1 Мастер создания MDL-файла

Для автоматизации процесса создания MDL-файла используется intellect_mdl.awx (см. каталог *Wizard*).

Создание MDL-файла с помощью Мастера выполняется следующим образом:

1. Поместить intellect_mdl.awx в каталог *Program Files|Microsoft Visual Studio|Common|MSDev98|Template*.
2. Запустить *Microsoft Visual C++*.
3. Создать новый проект **INTELLECT MDL WIZARD**.
4. Выполнить настройку проекта, следуя предложенным шагам.

В результате будет создан шаблон системного объекта. Проект будет включать все необходимые файлы, в том числе файл с описанием структуры объекта для intellect.dbi.

Внимание!

В настройках проекта требуется заменить расширение результирующего файла с dll на mdl.

Примечание.

При сборке проекта необходимо использовать **Release**.

2.2.6 Разработка RUN-файла

Управление устройствами выполняется через обмен сообщениями (командами) между RUN-файлом и ядром системы. Для реализации данного взаимодействия программного модуля с ядром используется *IIDK*, подробно рассмотренный в разделе *Intellect Integration Developer Kit (IIDK)*. Необходимую информацию можно также почерпнуть из исходных файлов демонстрационного модуля, которые прилагаются к документации.

Ниже приведен пример использования средств разработки *IIDK* для демонстрационного модуля *DEMO*.

```
CString port = "1100";

CString ip = "127.0.0.1";

CString id = "";

BOOL IsConnect = Connect (ip, port, id, myfunc);

if (!IsConnect)

{

// не удалось подключиться

AfxMessageBox("Error");

Return;

}

SendMsg(id,"CAM|1|REC"); // поставить камеру 1 на запись

SendMsg(id, "DEMO|1|RESTORE"); // восстановление связи с объектом DEMO

//включить устройство DEMO_DEVICE с адресом 1

SendMsg(id,"DEMO_DEVICE|1|ON|params<1>,param0_name<address>,param0_val<1>");

Disconnect(id);
```

Внимание!

Если создан mdl-файл, то для подключения к ядру ПК *Интеллект* объект **Интерфейс IIDK** в системе не создается. В качестве идентификатора подключения передается пустая строка, то есть id равен “”.

При выгрузке модуля ему посылается сообщение **WM_EXIT**:

#define WM_EXIT (WM_USER+2000)

Используя функцию WinAPI – *PostThreadMessage,* необходимо перехватить это сообщение и обеспечить корректную выгрузку модуля. В *VC++* и *MFC* сообщение **WM_EXIT** отлавливается в классе, наследуемом от *CWinApp*, в *Delphi* и *CBuilder – TApplication*.

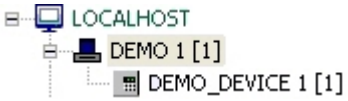
2.2.7 Создание и настройка интегрированных объектов (модулей) в ПК Интеллект

Для создания и настройки интегрированных объектов (модулей) в ПК *Интеллект* необходимо выполнить следующие действия:

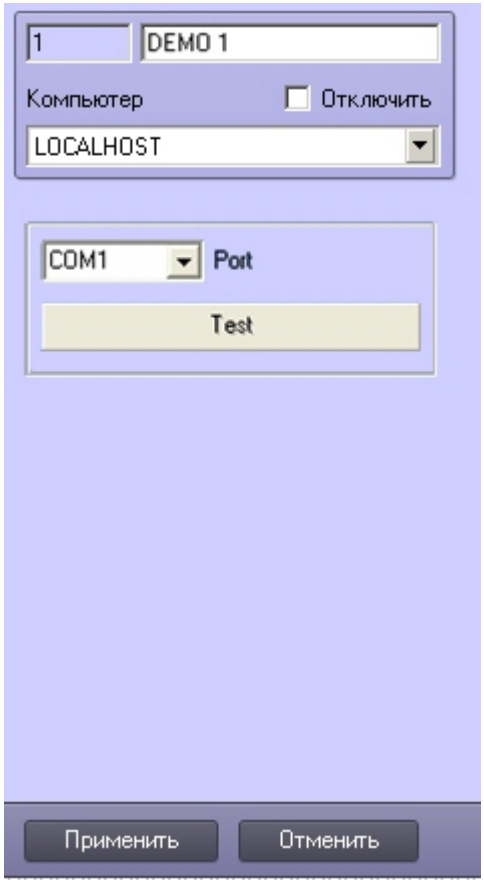
1. Разместить MDL и RUN-файлы в каталоге *Интеллект|Modules.*
2. Запустить ПК *Интеллект*.
3. На базе объекта **Компьютер** создать добавленные с помощью программного модуля объекты. Для демонстрационного модуля *DEMO* необходимо на базе объекта **Компьютер** создать объект **DEMO**.

   Примечание.

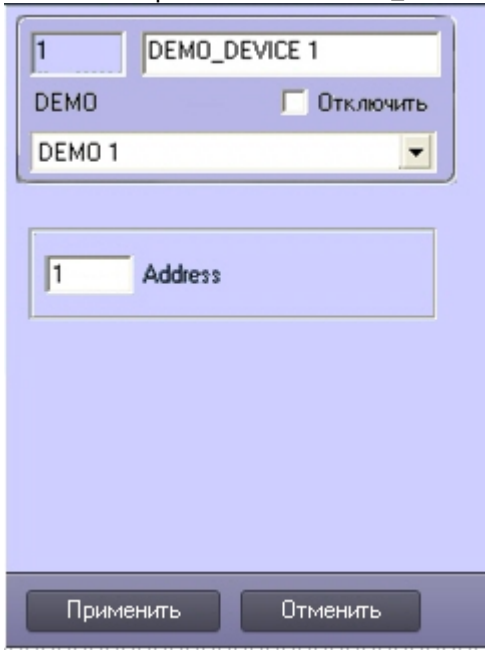
   На базе обекта **DEMO** создается дочерний объект **DEMO_DEVICE.**



В результате выполнения операции будут доступны панели настроек объектов.
Панель настроек объекта DEMO:



Панель настроек объекта DEMO_DEVICE:



4. Произвести настройку объектов.

Процесс создания и настройки интегрированных объектов в ПК *Интеллект* завершен.

3 Элемент управления ActiveX CamMonitor.ocx

3.1 Общее описание ActiveX-компонента CamMonitor.ocx

На странице:

3.1.1 Общее описание CamMonitor.ocx

CamMonitor.ocx представляет собой ActiveX компонент, который является полным аналогом интерфейсного объекта **Монитор видеонаблюдения**. Он позволяет управлять камерами, просматривать архив и т.д.

Компонент CamMonitor.ocx поддерживает работу в демо-режиме.

3.1.2 Требования к разработчику

Для использования CamMonitor.ocx требуется:

1. знание любого языка программирования, поддерживающего использование компонентов Component Object Model (COM);
2. знание основ программирования в Win32/Win64;
3. наличие среды разработки, поддерживающей работу с ocx-файлами.

Требования к программному обеспечению, используемому при интеграции

3.2 Установка CamMonitor.ocx

Внимание!

Не рекомендуется устанавливать ПК *Интеллект* и компонент CamMonitor.ocx на одном компьютере. Если это все же необходимо, то их версии должны совпадать, в противном случае работа компонента CamMonitor.ocx не гарантируется.

Установка CamMonitor.ocx осуществляется при помощи файла CamMonitorInstaller.exe, который располагается в папке <Директория установки *Интеллект*>\Redist\CamMonitor. Доступна как 32-битная, так и 64-битная версия данного компонента. Файлы установки компонентов разной битности расположены в соответствующих папках (x86 и x64 соответственно).

Если на компьютере установлен ПК *Интеллект*, то при установке файл 32-битного компонента CamMonitor.ocx помещается в папку <Директория установки ПК *Интеллект*>\Modules\, а файл 64-битного компонента устанавливается в папку <Директория установки ПК *Интеллект*>\Modules64\.

Если ПК *Интеллект* не установлен, то 32-битный компонент установится в папку c:\Program Files (x86)\ITV VideoPlayer\Modules\, а 64-битный – в папку c:\Program Files\ITV VideoPlayer x64\Modules64\

Также при установке осуществляется стандартная регистрация CamMonitor.ocx как компонента ActiveX.

CamMonitorInstaller.exe выполняет установку требуемых файлов для всех пользователей системы.

Помимо самой библиотеки устанавливается также пакет драйверов CodecPack и утилита ITV VideoPlayer, которая работает с использованием компонента CamMonitor.ocx и позволяет просматривать видеоархив по выбранной камере. Данная утилита по умолчанию устанавливается в папку C:\Program Files\ITV VideoPlayer\. Интерфейс утилиты схож с интерфейсом Converter.exe, однако в ней отсутствуют некоторые функции Converter.exe, не связанные с просмотром архива. Данную утилиту можно использовать для проверки корректности установки CamMonitor.ocx.

3.3 Параметры CamMonitor.ocx

На странице:

В этом разделе перечислены параметры, позволяющие настроить режим работы компонента CamMonitor: задать отображаемые элементы интерфейса, а также режим использования оверлея.

Все параметры представляют собой целое число типа long.

Значения параметров для настройки элементов интерфейса, перечисленные в таблицах, сформированы таким образом, чтобы в двоичном представлении числа была только одна единица. Чтобы задать значение параметра, необходимо при помощи операции исключающего ИЛИ (XOR) объединить требуемые значения параметров, получив таким образом некое число, разряды которого в двоичном представлении говорят о том, какие элементы интерфейса следует отображать, а какие нет. См. также Пример использования параметров.

Параметр OverlayMode отличается от прочих: он принимает значения от 0 до 2, и его значение задает режим использования оверлея.

3.3.1 CamMenuOptions

CamMenuOptions : long

Позволяет настроить функциональное меню камеры.

Допускается установка одного или нескольких флагов.

Возможные значения:

Значение	Описание
#define MENU_ENABLE_OPTION 0x00000001	Не используется. См. BUTTON_MENU_ENABLE_OPTION
#define MENU_ARM_ENABLE_OPTION 0x00000002	Отображать пункт меню **Поставить на охрану**
#define MENU_REC_ENABLE_OPTION 0x00000004	Отображать пункт меню **Начать запись**
#define MENU_CAMS_ENABLE_OPTION 0x00000008	Отображать пункт меню **Камера**
#define MENU_TITLES_ENABLE_OPTION 0x00000010	Отображать пункт меню **Отображение титров**

#define MENU_PROCESSING_ENABLE_OPTION 0x00000020	Отображать пункт меню **Обработка**
#define MENU_EXPORT_ENABLE_OPTION 0x00000040	Отображать пункт меню **Экспорт**

3.3.2 CamMenuProcessingOptions

CamMenuProcessingOptions : long

Позволяет настроить меню **Обработка** в функциональном меню камеры.

Допускается установка одного или нескольких флагов.

Возможные значения:

Значение	**Описание**
#define MENU_PROCESSING_DEINTERLACE_ENABLE_OPTION 0x00000001	Отображать пункт меню **Деинтерлейсинг**
#define MENU_PROCESSING_ZOOM_ENABLE_OPTION 0x00000002	Отображать пункт меню **Увеличение.** *Примечание. При сбросе данной опции пункт меню **Обработка->Увеличение** исчезает, но остаётся возможность увеличить изображение с помощью колёсика мыши.*
#define MENU_PROCESSING_CONTRAST_ENABLE_OPTION 0x00000004	Отображать пункт меню **Контрастирование**
#define MENU_PROCESSING_MASK_ENABLE_OPTION 0x00000008	Отображать пункт меню **Маска детектора**
#define MENU_PROCESSING_SHARP_ENABLE_OPTION 0x00000010	Отображать пункт меню **Резкость**

3.3.3 CamButtonsOptions

CamButtonsOptions : long

Позволяет настроить отображение кнопок компонента CamMonitor.

Допускается установка одного или нескольких флагов.

Возможные значения:

Значение	**Описание**
#define BUTTON_MODE_ENABLE_OPTION 0x00000100	Отображать кнопку входа в архив
#define BUTTON_TIME_ENABLE_OPTION 0x00000002	Отображать время
#define BUTTON_NAME_ENABLE_OPTION 0x00000004	Отображать название камеры
#define BUTTON_MENU_ENABLE_OPTION 0x00000008	Отображать кнопку вызова функционального меню
#define BUTTON_RAYS_ENABLE_OPTION 0x00000010	Не используется
#define BUTTON_MICS_ENABLE_OPTION 0x00000020	Не используется

3.3.4 MainPanelOptions

MainPanelOptions : long

Позволяет настроить отображение панели инструментов CamMonitor.

Допускается установка одного или нескольких флагов.

Возможные значения:

Значение	Описание
#define MAIN_PANEL_ENABLE_OPTION 0x00000001	Включает отображение панели
#define MAIN_PANEL_ENABLE_SCREENS_BUTTON 0x00000010	Отображать кнопку **Экраны** (см. Раскладка Окон видеонаблюдения на Мониторе видеонаблюдения).
#define MAIN_PANEL_ENABLE_BOOKMARK_BUTTON 0x00000020	Отображать кнопку **Создать закладку** (см. Создание закладок).
#define MAIN_PANEL_ENABLE_BOOKMARK_REVIEW_BUTTON 0x00000040	Отображать кнопку **Созданные закладки** (см. Список закладок).
#define MAIN_PANEL_ENABLE_AVIEXPORT_BUTTON 0x00000080	Отображать кнопку **Фоновый экспорт** (см. Утилита AviExport).

3.3.5 KeysOptions

KeysOptions : long

Позволяет настроить управление компонентом при помощи клавиатуры и мыши.

Допускается установка одного или нескольких флагов.

Возможные значения:

Значение	Описание
#define KEYS_ENABLE_OPTION 0x00000001	Включает возможность управления компонентом CamMonitor при помощи горячих клавиш, доступных для Монитора видеонаблюдения (см. Монитор видеонаблюдения).
#define TELEMETRY_DISABLE_OPTION 0x00000002	Отключает возможность управления телеметрией из компонента CamMonitor (см. Управление поворотными устройствами).
#define ARCH_DELETE_ENABLE_OPTION 0x00000004	Включает возможность удаления записей архива из списка видеозаписей (см. Удаление видеозаписей из архива).
#define ARCH_PROTECT_ENABLE_OPTION 0x00000008	Включает возможность защищать записи архива в списке видеозаписей (см. Защита записей и снятие защиты).

3.3.6 OverlayMode

OverlayMode : long

Задает режим отображения.

Возможные значения:

Значение	Описание
0	Не использовать Overlay

1	Overlay 1
2	Overlay 2

3.3.7 Пример использования параметров

```
DWORD options = CamMonitor1->CamMenuOptions;

options = options^MENU_CAMS_ENABLE_OPTION^MENU_ARM_ENABLE_OPTION^MENU_REC_ENABLE_OPTION;

CamMonitor1->CamMenuOptions = options;

CamMonitor1->CamMenuProcessingOptions ^= MENU_PROCESSING_MASK_ENABLE_OPTION;
```

3.4 Методы CamMonitor.ocx

На странице:

3.4.1 Connect

Connect(BSTR **ip**, BSTR **login**, BSTR **password**, BSTR **arch_password**, long **param**, long **port**) установка соединения с сервером/видеошлюзом/долговременным архивом.

- BSTR **ip** – IP адрес сервера.
- BSTR **login** – логин для соединения с сервером (может быть пустым).
- BSTR **password** – пароль на соединение с сервером (может быть пустым).
- BSTR **arch_password** – пароль для доступа к архиву (т.е. пароль администратора, может быть пустым).
- long **param** – роль, исполняемая сервером. Параметр является обязательным.
  - 0 – видеосервер;
  - 1 – оперативный архив;
  - 2 – видеошлюз.
- long **port** – задает порт подключения к серверу.
  - если передать 0, 1 или 2, то соединение будет устанавливаться с портом 900, 901 или 902 соответственно;
  - если передать 100, то будет выполняться соединение через порт 10504;

- если передать какое-либо другое значение, то будет осуществляться соединение по порту с номером port + 20000. Например, если port=900, то подключение происходит к порту сервера 20900.

Установка связи с сервером происходит **асинхронно**.

> **Внимание!**
>
> Если при вызове метода Connect() не указаны логин и пароль, то в элементе управления будет доступен просмотр видео со всех камер. Если разграничение прав важно в стороннем приложении, это следует учитывать на этапе разработки.

3.4.2 ShowCam

ShowCam(long **cam_id**, long **compress**, long **show**) выводит/скрывает камеру с экрана

- long **cam_id** – идентификатор(номер) камеры.
- long **compress** – уровень компрессии видео 0-5 (для локальной камеры =0). -1 - отображает в оригинальном формате транслируемом с камеры без рекомпресии.
- long **show** – флаг означающий действие: показать/скрыть камеру (1/0).

3.4.3 DoReactMonitor

DoReactMonitor(BSTR **react_string**) – управление поведением монитора/камер

- BSTR **react_string** – строковое представление реакции.

Пример формировании react_string:

```
react_string = "MONITOR||ARCH_FRAME_TIME|cam<3>,date<dd-mm-yy>,time<hh:mm:ss>,mode<1>";

CamMonitor1->DoReactMonitor(react_string);
```

Результат вызова функции с таким параметром: камера 3 будет переведена в режим архива видеосервера, и архив будет позиционирован на дату «dd-mm-yy» и время «hh:mm:ss» (дату и время необходимо задавать только в таком формате). Параметр mode может принимать следующие значения:

0 – видеошлюз, если задан (если не задан, то видеосервер).

1 – видеосервер.

2 – долговременный архив.

“MONITOR|**<id игнорируется >**|ARCH_FRAME_TIME|...”

> **Примечание:**
>
> Опционально можно указать также точность при позиционировании до миллисекунд например:
>
> DoReactMonitor(“MONITOR||ARCH_FRAME_TIME|cam<3>,date<02-10-05>,time<12:12:22.345>,mode<1>")

Пример. Вывод 2-го потока камеры 14 на монитор 1:

"MONITOR|1|ADD_CAM|cam<14>,cam_id<14>,compress<1>,stream_id<14.2>"

Пример. Задание частоты кадров равной 1 при просмотре архива с камеры 11

"MONITOR||CAM_PARAMS|cam<11>;arch_fps<1>"

3.4.4 RemoveAllCams

RemoveAllCams() : long – удаление всех камер с экрана

3.4.5 IsConnected

IsConnected() : boolean - метод говорит о наличии/отсутствии связи с видеосервером.

3.4.6 GetCurIp

GetCurIp() : BSTR – возвращает IP адрес сервера, ранее указанный при вызове **Connect**.

3.4.7 SendRawMessage

SendRawMessage(BSTR **msg**) – дать видеосерверу команду на исполнение.

- BSTR **msg** – строковое представление команды.

Примеры вызова функции:

```
m_Cam.SendRawMessage("CAM|1|REC");
```

```
m_Cam.SendRawMessage("CAM|1|REC_STOP");
```

```
m_Cam.SendRawMessage("CAM|1|ARM");
```

```
m_Cam.SendRawMessage("CAM|1|DISARM");
```

3.4.8 Disconnect

Disconnect() – осуществляет отсоединение от видеосервера.

3.4.9 SetCallBackOptions

SetCallBackOptions(int **cam_id**, int **options**) – задает параметры получения видеоизображения с камеры.

- int **cam_id** – идентификатор(номер) камеры.
- int **options** – опции. Возможные значения параметра **options:**
  - WithoutVideoFrame = 0x00 – не присылать кадры из модуля видео.
  - WithVideoFrame = 0x01 – присылать кадры из модуля видео.
  - WithExtendedParams = 0x02 – получать видео кадры с дополнительными параметрами (время, число кадров, субтитры).
  - WithInformationLayout = 0x04 – отображать видео в окне с элементами управления (контекстное меню).
  - WithCompressedData = 0x08 – отображать видео в исходном формате без распаковки (если оно есть).
  - WithoutDecode = 0x10 – отключить декодирование видео на сервере.
  - WithoutSubtitles=0x20 – отключить субтитры.

> **Примечание.**
>
> Параметр options формируется так же, как параметры компонента CamMonitor.ocx – см. Параметры CamMonitor.ocx

3.4.10 SetParam

SetParam(BSTR **param_name**, BSTR **param_value**) - задает количество окон видеокамер в CamMonitor.

- BSTR **param_name** - строковое представление длины или ширины.
- BSTR **param_value** - количество окон видеокамер.

Примеры вызова функции:

```
m_cam.SetParam("monitor_ch", m_NH);
m_cam.SetParam("monitor_cw", m_NW);
```

3.5 События CamMonitor.ocx

OnCamListChange (long **cam_id**, long **action**) - возникает при установлении связи с сервером или при изменении количества камер на сервере

• long **cam_id** – идентификатор камеры.

• long **action** – равен 1, если камера с **id** == **cam_id** существует, иначе **action** == **0.**

Данное событие возникает столько раз, сколько камер на данном видеосервере. Признаком окончания вызовов **OnCamListChange** является отрицательное значение параметра **cam_id** (cam_id < 0).

Если на сервере находится, например, 3 камеры (1, 2, 3), то последовательно возникнут события:

CamListChange(1,1)

CamListChange(2,1)

CamListChange(3,1)

CamListChange(-1,1)

Пример:

Показать камеру с cam_id =2 с уровнем компресии compress =1;

```
CamMonitor1CamListChange(long cam_id, long action)
{
        if(cam_id == -1)
        {
                CamMonitor1->ShowCam(2,1,1);
        }
}
```

4 HTTP API ПК Интеллект

4.1 Общие сведения о HTTP API

Программно HTTP API предоставляется модулем web2 (*Веб-сервер 2.0*).

> **Примечание.**
>
> См. Руководство Администратора, раздел Настройка Сервера для подключения Клиентов с помощью модуля Веб-сервер 2.0.

HTTP API позволяет использовать следующие функции:

1. Получение сведений об интерактивных картах: списка карт, имени карты, списка слоев карты, параметров слоя, фонового рисунка слоя, информации о списке точек и отдельной точке на слое (см. Карта).
2. Получение сведений о классах объектов, созданных на Сервере, списка состояний для класса объектов и информации о состоянии, иконки для определенного состояния (см. Классы объектов).
3. Получение списка объектов, созданных на сервере, информации об отдельном объекте, состояния объекта, списка доступных действий с объектом (см. Объекты).
4. Получение событий с Сервера как отдельно, так и блоками (см. Получение событий).
5. Отсылать команды на Сервер (см. Отсылка команд на сервер).
6. Запускать выполнение макрокоманд (см. Макрокоманды).
7. Работать с видео: получить кадры, запрашивать конфигурацию, получать живое видео и архив, управлять записью, ставить и снимать камеры с охраны, управлять телеметрией (см. Видео).
8. Использовать системы нотификации для подписки приложения на сообщения APNS (см. Нотификация).
9. Получать живой и архивный звук (см. Звук).
10. Отправлять события и реакции в ядро ПК Интеллект (см. Отправка реакций и событий в ПК Интеллект по HTTP-запросу).

В примерах, приводимых в данном разделе, используются следующие обозначения:

- Port – порт. По умолчанию порт модуля **Веб-сервер 2.0** — 8085. Указывать порт при отправке команд HTTP API **обязательно**.
- /web2 – веб-контекст, в котором работает модуль web2. Это контекст веб-приложения.

Далее описание будет опускаться там, где действие запроса понятно из контекста.

> **Внимание!**
>
> URL, id объектов и расширения файлов чувствительны к регистру.

> **Примечание.**
>
> Дата и время указываются в формате RFC3339, подробнее см. http://www.ietf.org/rfc/rfc3339.txt

4.1.1 Формат ответа по умолчанию

По умолчанию формат ответа JSON. Включение ответа по умолчанию в формате XML осуществляется на панели настройки объекта **Веб-сервер 2.0** (см. Настройка типа ответа на запросы HTTP API по умолчанию). Также формат ответа может быть явно указан в заголовке **Accept**, например **application/json** или **application/xml**.
Указанный формат ответа в запросе имеет больший приоритет, чем заданный на панели настройки объекта **Веб-сервер 2.0** формат ответа по умолчанию.

4.1.2 Кросс-доменные запросы (CORS)

Для выполнения кросс-доменных запросов или для получения доступа к необходимым заголовкам в ответе (например, в связи с ограничениями из-за политики CORS браузера), необходимо в заголовке запроса указать **Origin** (домен сайта, с которого происходит запрос). В таком случае в ответе будет содержаться заголовок **Access-Control-**

Allow-Origin, который указывает на разрешение доступа к ресурсу с указанного домена кросс-сайтовым способом. Заголовок **Access-Control-Allow-Origin: *** означает, что к ресурсу можно получить доступ с любого домена кросс-сайтовым способом.

4.2 Версия продукта

4.2.1 Общий формат запроса:

```
GET http://IP-адрес:порт/web2/product/version
```

4.2.2 Пример запроса:

GET http://127.0.0.1:8085/web2/product/version

4.2.3 Пример ответа:

В ответ приходит text/plain строка вида:

```
Intellect/4.11.2.2875
```

Это означает, что сервер поддерживает протокол, описанный в данном документе. Строка может меняться в зависимости от версии продукта. Это позволяет различать два схожих по функциональным возможностям, но разных по протоколу веб-сервера в разных продуктах.

4.3 Авторизация в ПК Интеллект по token ключу

Авторизация в ПК *Интеллект* по token ключу позволяет:

- В url-запросе вместо указания параметров "login" и "password" указывать токен в параметре "token". Пример запроса видео с авторизацией в ПК *Интеллект* по token ключу:

  ```
  http://127.0.0.1:80/video/action.do?normalize=true&version=4.10.0.0&video_in=CAM:1&token=EoHWC_zXFILImB0hL4QgjPc5624cJXMF
  ```

- Использовать в заголовке запроса в параметре "Authorization" как Bearer Token Authentication. Например:

  ```
  Authorization: Bearer PJ_eHSwUsqjXX7PRZMB8hm_zKEnCg3hE"
  ```

4.3.1 Общий формат запроса:

GET/POST http://{login}:{password}@IP-адрес:порт/token?expires_in={expires_in}

4.3.2 Параметры запроса:

Параметр	Обязательный	Описание
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

Параметр	Обязательный	Описание
expires_in	Нет	Время действия токена в секундах. Максимальное значение - 1 день. Токен перестает действовать через указанный период времени Значение по-умолчанию: "**1800**". Для того, чтобы разлогиниться - указать значение: "**0**" *Примечание. Для одного пользователя может существовать только 1 токен.*

4.3.3 Пример запроса:

```
GET/POST http://USER:PASSWORD@127.0.0.1:8085/token?expires_in=1800
```

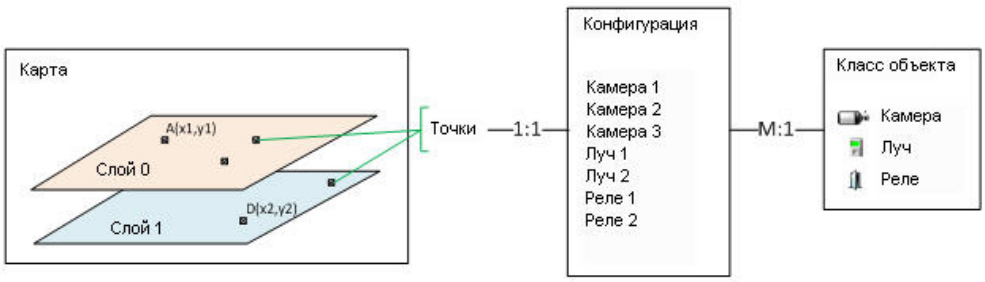
4.3.4 Пример ответа:

```
{
"access_token":"PJ_eHSwUsqjXX7PRZMB8hm_zKEnCg3hE"
"token_type":"bearer"
"expires_in":"1800"
}
```

4.3.5 Параметры ответа:

Параметр	Описание
access_token	Токен
token_type	Тип токена
expires_in	Время действия токена в секундах

4.4 Карта



На Сервере может быть создано несколько карт. Каждая карта может содержать один и более слоёв. На каждом слое расположены точки. Каждая точка связана с одним из объектов конфигурации.

Конфигурация – это объекты ПК *Интеллект*. Каждый объект является объектом определённого класса. Каждый объект имеет одно состояние и список действий, которые можно с ним производить.

Класс объекта описывает его вид (значки), возможные состояния и возможные действия с объектом в каждом из состояний .

4.4.1 Получение списка карт

4.4.1.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/

4.4.1.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/

4.4.1.3 **Пример ответа:**


```
<kartas>
   <karta>
      <id>2</id>
      <layers>
         <layer>
            <geo_h>10.0</geo_h>
            <height>578</height>
            <id>2</id>
            <lat_bl>43.5905</lat_bl>
            <lat_br>43.6875</lat_br>
            <lat_c>43.6395</lat_c>
            <lat_tl>43.5914</lat_tl>
            <lat_tr>43.6885</lat_tr>
            <lon_bl>43.4584</lon_bl>
            <lon_br>43.4599</lon_br>
            <lon_c>43.4809</lon_c>
            <lon_tl>43.5018</lon_tl>
            <lon_tr>43.5033</lon_tr>
            <mapId>2</mapId>
            <name>Layer 2</name>
            <points>
               <point>
                  <id>CAM:1</id>
                  <layerId>2</layerId>
                  <mapId>2</mapId>
                  <angle>0.0</angle>
                  <geo_angle>0.0</geo_angle>
                  <latitude>43.47727</latitude>
                  <longitude>43.602381</longitude>
                  <x>95.0</x>
                  <y>329.0</y>
               </point>
            </points>
            <width>800</width>
            <zoomDef>1.0</zoomDef>
            <zoomMax>4.0</zoomMax>
            <zoomMin>0.25</zoomMin>
            <zoomStep>0.25</zoomStep>
         </layer>
      </layers>
      <name>Map 2</name>
   </karta>
</kartas>
```

4.4.1.4 Параметры ответа

Параметр	Описание
Параметры группы <karta>	
id	Идентификатор карты
name	Название карты
layers	Список слоев
Параметры группы <layer>	
geo_h	Метка высоты (см. Настройка привязки карты к координатной сетке)
height	Высота слоя карты в пикселях
width	Ширина слоя карты в пикселях
id	Идентификатор слоя
lat_bl	Широта: нижний левый угол
lat_br	Широта: нижний правый угол
lat_c	Широта: центр
lat_tl	Широта: верхний левый угол
lat_tr	Широта: верхний правый угол
lon_bl	Долгота: нижний левый угол
lon_br	Долгота: нижний правый угол
lon_c	Долгота: центр
lon_tl	Долгота: верхний левый угол
lon_tr	Долгота: верхний правый угол
mapId	Идентификатор карты
name	Название слоя
points	Список точек на слое
zoomDef	Масштаб по умолчанию

Параметр	Описание
zoomMax	Минимальный масштаб
zoomMin	Максимальный масштаб
zoomStep	Шаг масштабирования
Параметры группы <point>	
id	Тип и идентификатор объекта в формате ТИП:ID
layerId	Идентификатор слоя
mapId	Идентификатор карты
angle	Угол поворота значка объекта
geo_angle	Угол обзора (для камеры, см. Настройка отображения угла обзора камеры на Карте)
latitude	Широта (координата точки)
longitude	Долгота (координата точки)
x	Координата точки относительно подложки по оси X
y	Координата точки относительно подложки по оси Y

4.4.2 Информация об одной карте

4.4.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/

4.4.2.2 Параметры запроса:

Параметр	Обязательный	Описание
plan	Да	Идентификатор карты

4.4.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/2

4.4.2.4 Пример ответа:

```
<karta>
  <id>2</id>
  <name>This is plan of a building</name>
</karta>
```

4.4.2.5 Параметры ответа:

Параметр	Описание
id	Идентификатор карты
name	Название карты

4.4.3 Список слоёв для выбранной карты

4.4.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/layers

4.4.3.2 Параметры запроса

Параметр	Обязательный	Описание
plan	Да	Идентификатор карты

4.4.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/2/layers

4.4.3.4 Пример ответа:

```
<layers>
   <layer>
      <geo_h>10.0</geo_h>
      <height>578</height>
      <id>2</id>
      <lat_bl>43.5905</lat_bl>
      <lat_br>43.6875</lat_br>
      <lat_c>43.6395</lat_c>
      <lat_tl>43.5914</lat_tl>
      <lat_tr>43.6885</lat_tr>
      <lon_bl>43.4584</lon_bl>
      <lon_br>43.4599</lon_br>
      <lon_c>43.4809</lon_c>
      <lon_tl>43.5018</lon_tl>
      <lon_tr>43.5033</lon_tr>
      <mapId>2</mapId>
      <name>Layer 2</name>
      <points>
         <point>
            <id>CAM:1</id>
            <layerId>2</layerId>
            <mapId>2</mapId>
            <angle>0.0</angle>
            <geo_angle>0.0</geo_angle>
            <latitude>43.47727</latitude>
            <longitude>43.602381</longitude>
            <x>95.0</x>
            <y>329.0</y>
         </point>
      </points>
      <width>800</width>
      <zoomDef>1.0</zoomDef>
      <zoomMax>4.0</zoomMax>
      <zoomMin>0.25</zoomMin>
      <zoomStep>0.25</zoomStep>
   </layer>
</layers>
```

4.4.3.5 Параметры ответа

Параметр	Описание
Параметры группы <layer>	
geo_h	Метка высоты (см. Настройка привязки карты к координатной сетке)
height	Высота слоя карты в пикселях
width	Ширина слоя карты в пикселях
id	Идентификатор слоя

Параметр	Описание
lat_bl	Широта: нижний левый угол
lat_br	Широта: нижний правый угол
lat_c	Широта: центр
lat_tl	Широта: верхний левый угол
lat_tr	Широта: верхний правый угол
lon_bl	Долгота: нижний левый угол
lon_br	Долгота: нижний правый угол
lon_c	Долгота: центр
lon_tl	Долгота: верхний левый угол
lon_tr	Долгота: верхний правый угол
mapId	Идентификатор карты
name	Название слоя
points	Список точек на слое
zoomDef	Масштаб по умолчанию
zoomMax	Минимальный масштаб
zoomMin	Максимальный масштаб
zoomStep	Шаг масштабирования
Параметры группы <point>	
id	Тип и идентификатор объекта в формате ТИП:ID
layerId	Идентификатор слоя
mapId	Идентификатор карты
angle	Угол поворота значка объекта
geo_angle	Угол обзора (для камеры, см. Настройка отображения угла обзора камеры на Карте)
latitude	Широта (координата точки)

Параметр	Описание
longitude	Долгота (координата точки)
x	Координата точки относительно подложки по оси X
y	Координата точки относительно подложки по оси Y

4.4.4 Информация о конкретном слое

4.4.4.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/layers/{base}/

4.4.4.2 Параметры запроса:

Параметр	Обязательный	Описание
plan	Да	Идентификатор карты
base	Да	Идентификатор слоя

4.4.4.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/2/layers/2/

4.4.4.4 **Пример ответа:**

```
<layer>
    <height>1000</height>
    <id>2</id>
    <mapId>2</mapId>
    <name>Base layer for plan</name>
    <width>1000</width>
    <zoomDef>1.0</zoomDef>
    <zoomMax>4.0</zoomMax>
    <zoomMin>0.25</zoomMin>
    <zoomStep>0.25</zoomStep>
</layer>
```

4.4.4.5 Параметры ответа:

Параметр	Описание
height	Высота слоя карты в пикселях
width	Ширина слоя карты в пикселях
id	Идентификатор слоя

Параметр	Описание
mapId	Идентификатор карты
name	Название слоя
zoomDef	Масштаб по умолчанию
zoomMax	Минимальный масштаб
zoomMin	Максимальный масштаб
zoomStep	Шаг масштабирования

4.4.5 Фоновый рисунок слоя

4.4.5.1 Общий формат запроса

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/layers/{base}/image.{ext}

4.4.5.2 Параметры запроса

Параметр	Обязательный	Описание
plan	Да	Идентификатор карты
base	Да	Идентификатор слоя
ext	Да	Расширение файла. Допустимые значения: **png** или **jpg**

4.4.5.3 Пример запроса

GET http://127.0.0.1:8085/web2/secure/kartas/2/layers/3/image.png

4.4.5.4 Пример ответа

В ответ приходит изображение в указанном формате.

4.4.5.5 Ошибки выполнения запроса

Ошибка	Описание
404	Запрошено изображение в формате **JPEG**

4.4.6 Список точек на слое

4.4.6.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/layers/{base}/points/

4.4.6.2 Параметры запроса:

Параметр	Обязательный	Описание
plan	Нет	Идентификатор карты
base	Нет	Идентификатор слоя

4.4.6.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/2/layers/2/points/

4.4.6.4 Пример ответа:

> **Примечание.**
>
> Если тип отображения добавленного объекта на карте отличен от **Изображение**, то на запрос приходит пустой ответ. См. также Прикрепление объектов к слою интерактивной карты

```
<points>
  <point>
    <id>CAM:1</id>
    <layerId>2</layerId>
    <mapId>2</mapId>
    <angle>0.0</angle>
    <geo_angle>0.0</geo_angle>
    <latitude>43.47727</latitude>
    <longitude>43.602381</longitude>
    <x>95.0</x>
    <y>329.0</y>
  </point>
</points>
```

4.4.6.5 Параметры ответа:

Параметр	Описание
id	Идентификатор объекта в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
layerId	Идентификатор слоя
mapId	Идентификатор карты
angle	Угол поворота значка объекта
geo_angle	Угол обзора (для камеры, см. Настройка отображения угла обзора камеры на Карте)
latitude	Широта (координата точки)
longitude	Долгота (координата точки)
x	Координата X верхнего левого угла значка объекта

Параметр	Описание
y	Координата Y верхнего левого угла значка объекта

Координатная сетка привязана к слою следующим образом:

Т.е. x и y не могут быть отрицательными, но могут быть дробными.

4.4.7 Информация об отдельной точке на слое

4.4.7.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/kartas/{plan}/layers/{base}/points/{CAM:id}

4.4.7.2 Параметры запроса:

Параметр	Обязательный	Описание
plan	Нет	Идентификатор карты
base	Нет	Идентификатор слоя
CAM:id	Да	Идентификатор объекта в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"

4.4.7.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/kartas/plan/layers/base/points/CAM:2

4.4.7.4 Пример ответа:

```
<point>
    <id>CAM:2</id>
    <layerId>base</layerId>
    <mapId>plan</mapId>
    <x>200.0</x>
    <y>200.0</y>
</point>
```

4.4.7.5 Параметры ответа:

Параметр	Описание
id	Идентификатор объекта в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
layerId	Идентификатор слоя
mapId	Идентификатор карты
x	Координата X верхнего левого угла значка объекта
y	Координата Y верхнего левого угла значка объекта

4.5 Классы объектов

4.5.1 Список классов объектов, которые существуют на сервере

4.5.1.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses

4.5.1.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses

4.5.1.3 Пример ответа:

```
<objectClasses>
    <objectClass>
        <id>GRELE</id>
    </objectClass>
    <objectClass>
        <id>USERS</id>
    </objectClass>
    <objectClass>
        <id>CAM</id>
    </objectClass>
    <objectClass>
        <id>RIGHTS</id>
    </objectClass>
    <objectClass>
        <id>GRAY</id>
    </objectClass>
</objectClasses>
```

4.5.1.4 Параметры ответа:

Параметр	Описание
id	Идентификатор класса объектов

4.5.2 Отдельный класс объектов

4.5.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses/{objectClass}/

4.5.2.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Идентификатор класса объектов

4.5.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses/GRELE/

4.5.2.4 Пример ответа:

```
<objectClass>
    <id>GRELE</id>
</objectClass>
```

4.5.2.5 Параметры ответа:

Параметр	Описание
id	Идентификатор класса объектов

4.5.3 Список состояний для определённого класса объектов

4.5.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses/{objectClass}/states/

4.5.3.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Идентификатор класса объектов

4.5.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses/GRELE/states/

4.5.3.4 Пример ответа:

```
<states>
    <state>
        <id>off</id>
    </state>
    <state>
        <id>on</id>
    </state>
    <state>
        <id>disabled</id>
    </state>
</states>
```

4.5.3.5 Параметры ответа:

Параметр	Описание
id	Идентификатор всех возможных состояний класса объектов

4.5.4 Информация о конкретном состоянии

4.5.4.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses/{ObjectClass}/states/{State}/

4.5.4.2 Параметры запроса:

Параметр	Обязательный	Описание
ObjectClass	Да	Идентификатор класса объектов
State	Да	Идентификатор состояния класса объектов

4.5.4.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses/GRELE/states/off/

4.5.4.4 Пример ответа:

```
<state>
    <id>off</id>
</state>
```

4.5.4.5 Параметры ответа:

Параметр	Описание
id	Идентификатор состояния класса объектов

4.5.5 Получение иконки для определённого состояния

4.5.5.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses/{ObjectClass}/states/{State}/image.png

4.5.5.2 Параметры запроса:

Параметр	Обязательный	Описание
ObjectClass	Да	Название класса объектов
State	Да	Идентификатор состояния класса объектов

4.5.5.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses/GRELE/states/off/image.png

4.5.5.4 Пример ответа:

В ответ придет изображение в формате png.

4.5.6 Список событий для определенного класса объектов

4.5.6.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/objectClasses/{ObjectClass}/events/

4.5.6.2 Параметры запроса:

Параметр	Обязательный	Описание
ObjectClass	Да	Идентификатор класса объектов

4.5.6.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/objectClasses/GRELE/events/

4.5.6.4 Пример ответа:

```
<events>
    <event>
        <id>23</id>
        <sid>grele.disable</sid>
        <description>Disable rele</description>
    </event>
    <event>
        <id>24</id>
        <sid>grele.enable</sid>
        <description>Enable rele</description>
    </event>
</events>
```

4.5.6.5 Параметры ответа:

Параметр	Описание
id	Идентификатор объекта
sid	Команда события
description	Описание события

4.6 Объекты

4.6.1 Получение списка всех объектов сервера

4.6.1.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/configuration?pageItems={pageItems}&page={page}

4.6.1.2 **Параметры запроса:**

Параметр	Обязательный	Описание
pageItems	Нет	Задает номер страницы, отображаемой в результате запроса . pageItems > 0. По умолчанию pageItems=1.
page	Нет	Задает количество объектов, выводимых на странице. page > 0. По умолчанию page=1000.

> **Внимание!**
>
> Если в системе много объектов (>1000) необходимо использовать постраничный вывод.

Обработка всех объектов производится перебором страниц, до получения пустого массива.

4.6.1.3 **Пример запроса:**

GET http://127.0.0.1:8085/web2/secure/configuration

4.6.1.4 **Пример ответа:**

Запрос возвращает список следующих объектов с указанием состояния:

- список камер, добавленных в Web-сервер, с указанием идентификаторов соответствующих микрофонов, динамиков, поворотного устройства, пресетов (см. также Выбор и настройка видеокамер для Web-сервера);
- список камер, добавленных на карты, выбранные для использования *Веб-сервером 2.0* (см. Выбор карт);
- список лучей;
- список реле;
- список макрокоманд;
- список RTSP-серверов с указанием используемых портов и добавленных в них камер;
- список областей и разделов.

JSON:

```
[
    {
        "id": "1",
        "name": "Область 1",
        "regions": [
            {
                "id": "1.1",
                "zoneId": "1",
                "name": "Раздел 1.1",
                "zoneDescription": "Описание зоны"
            }
        ]
    },
    {
        "id": "2",
        "name": "Область 2",
        "regions": [
            {
                "id": "2.1",
                "zoneId": "2",
                "name": "Раздел 2.1",
                "zoneDescription": "Описание зоны"
            }
        ]
    }
]
```


XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<baseObjects>
    <CAM>
        <displayId>1</displayId>
        <displayName>Экран 1</displayName>
        <extId>5</extId>
        <id>CAM:5</id>
        <monitorId>1</monitorId>
        <monitorName>Монитор 1</monitorName>
        <name>Камера 5</name>
        <regionDescription>Описание области</regionDescription>
        <regionId>2.1</regionId>
        <regionName>Раздел 2.1</regionName>
        <state>
            <fullState>DISARMED</fullState>
            <id>connected</id>
            <type>NORMAL</type>
        </state>
        <type>CAM</type>
        <zoneId>2</zoneId>
        <zoneName>Область 2</zoneName>
        <additionalInfo></additionalInfo>
        <micId></micId>
        <presets/>
        <speakerId></speakerId>
        <telemetryId></telemetryId>
    </CAM>
</baseObjects>
```

4.6.1.5 Параметры ответа:

Параметр	Описание
Общие параметры	
extId	Идентификатор объекта
id	Тип и идентификатор объекта в формате ТИП:ID
name	Название объекта
state	Состояние объекта. В параметрах <id> и <type> указано состояние в терминах API, подробнее см. Состояние отдельного объекта
type	Тип объекта
Особые параметры	
cams	Список камер в RTSP-сервере через точку с запятой
port	Порт, используемый RTSP-сервером
regionDescription	Описание области

Параметр	Описание
regionId	Идентификатор раздела, в который добавлен объект
regionName	Название раздела
zoneId	Идентификатор области, в который добавлен объект
zoneName	Название области
latitide	Широта (координата объекта при наличии геопривязки).
longitude	Долгота (координата объекта при наличии геопривязки).
monitorId	Для камеры: идентификатор монитора, на который добавлена камера.
monitorName	Для камеры: название монитора, на который добавлена камера.
geo_angle	Для камеры: угол обзора (см. Настройка отображения угла обзора камеры на Карте)
additionalInfo	Для камеры: значение поля **Дополнительная информация**
micId	Для камеры: идентификатор связанного микрофона.
presets	Для камеры: список предустановок.
speakerId	Для камеры: идентификатор связанного динамика.
telemetryId	Для камеры: идентификатор устройства управления телеметрией.
displayId	Для интерфейсных объектов: идентификатор экрана.
displayName	Для интерфейсных объектов:название экрана.

4.6.2 Информация об отдельном объекте

4.6.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/configuration/{objectClass}:{id}/

4.6.2.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Идентификатор класса объектов
id	Да	Идентификатор объекта

4.6.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/configuration/GRAY:2/

4.6.2.4 Пример ответа:

```
<GRAY>
    <id>GRAY:2</id>
    <name>Луч 2</name>
    <state>
        <id>alarmed</id>
    </state>
</GRAY>
```

4.6.2.5 Параметры ответа:

Параметр	Описание
id	Идентификатор объекта в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
name	Название объекта в ПК *Интеллект*
state id	Текущее состояние объекта

4.6.3 Состояние отдельного объекта

4.6.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/configuration/{objectClass}:{id}/state/

4.6.3.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Идентификатор класса объектов
id	Да	Идентификатор объекта

4.6.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/configuration/GRAY:2/state/

4.6.3.4 Пример ответа:

```
<state>
        <fullState>ON,ARMED</fullState>
        <id>armed</id>
        <type>NORMAL</type>
</state>
```

4.6.3.5 Параметры ответа:

Параметр	Описание
fullState	Полное состояние объекта, хранящееся в базе данных
id	Состояние объекта в терминах HTTP API
type	Состояние объекта в терминах HTTP API

Возможные значения данных параметра fullState для Луча:

Состояние луча	**fullState в web запросе**	dbo.state
Поставлен на охрану + замкнут	ON,ARMED	ON|ARMED
Поставлен на охрану + замкнут + тревога	ON,ALARMED	ON|ALARMED
Поставлен на охрану + замкнут + тревога принята	ON,CONFIRMED	ON|CONFIRMED
Поставлен на охрану + замкнут + обрыв связи	ON,DETACHED_DISARM	ON|DETACHED_DISARM
Снят с охраны + замкнут	ON,DISARMED	ON|DISARMED
Снят с охраны + замкнут + тревога	ON,ALARMED	ON|ALARMED
Снят с охраны + замкнут + тревога принята	ON,CONFIRMED	ON|CONFIRMED
Снят с охраны + замкнут + обрыв связи	ON,DETACHED_DISARM	ON|DETACHED_DISARM
Поставлен на охрану + разомнут	ARMED,OFF	ARMED|OFF
Поставлен на охрану + разомнут + тревога	OFF,ALARMED	OFF|ALARMED
Поставлен на охрану + разомнут + тревога принята	OFF,CONFIRMED	OFF|CONFIRMED
Поставлен на охрану + разомнут + обрыв связи	DETACHED_DISARM,OFF	DETACHED_DISARM|OFF
Снят с охраны + разомнут	DISARMED,OFF	DISARMED|OFF
Снят с охраны + разомнут + тревога	OFF,ALARMED	OFF|ALARMED
Снят с охраны + разомнут + тревога принята	OFF,CONFIRMED	OFF|CONFIRMED
Снят с охраны + разомнут + обрыв связи	DETACHED_DISARM,OFF	DETACHED_DISARM|OFF

4.6.4 Список доступных действий с объектом, находящимся в определённом состоянии

Список действий запрашивается не по классу объекта, а берётся из контекста конкретного объекта, т.к. возможны различные права пользователя на объекты одного и того же класса. Работа с полученным списком описана в разделе Отсылка команд на сервер.

4.6.4.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/configuration/{objectClass}:{id}/state/actions/

4.6.4.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Название класса объектов
id	Да	Идентификатор объекта

4.6.4.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/configuration/GRAY:2/state/actions/

4.6.4.4 Пример ответа:

```
<actions>
    <action>
        <description>Disarm</description>
        <hidden>false</hidden>
        <id>DISARM</id>
    </action>
    <action>
        <description>Arm</description>
        <hidden>false</hidden>
        <id>ARM</id>
    </action>
    <action>
        <description>Classify alarm</description>
        <hidden>false</hidden>
        <id>CONFIRM</id>
    </action>
</actions>
```

Если состояние объекта не предусматривает никаких действий, то xml будет таким:

```
<actions/>
```

4.6.4.5 Параметры ответа:

Параметр	Описание
description	Текстовое описание реакции
hidden	true – реакция не отображается в интерфейсе (на карте, в макрокомандах и т.д.) false – реакция отображается в интерфейсе
id	Системное название реакции

4.6.5 Получение списка всех областей и регионов

4.6.5.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/configuration/zones

4.6.5.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/configuration/zones

4.6.5.3 Пример ответа:

JSON:

```
[
    {
        "id": "1",
        "name": "Область 1",
        "regions": [
            {
                "id": "1.1",
                "zoneId": "1",
                "name":"Раздел 1.1",
                "zoneDescription": "Описание области"
            }
        ]
    },
    {
        "id": "2",
        "name": "Область 2",
        "regions": [
            {
                "id": "2.1",
                "zoneId": "2",
                "name": "Раздел 2.1",
                "zoneDescription": "Описание области"
            }
        ]
    }
]
```

XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<zones>
    <zone>
        <id>1</id>
        <name>Область 1</name>
        <regions>
            <id>1.1</id>
            <name>Раздел 1.1</name>
            <zoneDescription>Описание области</zoneDescription>
            <zoneId>1</zoneId>
        </regions>
    </zone>
    <zone>
        <id>2</id>
        <name>Область 2</name>
        <regions>
            <id>2.1</id>
            <name>Раздел 2.1</name>
            <zoneDescription>Описание области</zoneDescription>
            <zoneId>2</zoneId>
        </regions>
    </zone>
</zones>
```

4.6.5.4 Параметры ответа:

Параметр	Описание
id	Идентификатор области/раздела
name	Название области/раздела в ПК *Интеллект*
zoneDescription	Описание области

4.7 Получение событий

Соединение не разрывается и события приходят бесконечно.

4.7.1 Общий формат запроса:

http://IP-адрес:порт/web2/secure/feed/

4.7.2 Пример запроса:

http://127.0.0.1:8085/web2/secure/feed/

4.7.3 Пример ответа:

```
<message>
    <action>update</action>
    <objectId>CAM:1</objectId>
    <state>disconnected</state>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <x>10.0</x>
    <y>123.9</y>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <state>connected</state>
    <x>300.8</x>
    <y>670</y>
</message>

<message>
    <action>state</action>
    <objectId>CAM:1</objectId>
    <x>100</x>
    <y>100</y>
</message>

<message>
    <action>ping</action>
</message>
```

4.7.4 Параметры ответа:

Параметр	Описание
action	Тип события. Возможные значение: create, delete, update.
objectId	id объекта, от которого приходит событие (обязательно приходит с update, delete, create).
state	id нового состояния объекта (обязательно приходит в create. Если состояние не изменилось, то в событии update состояния не будет).
x, y	Новые координаты, если изменились.

4.7.5 Получение событий видеоподсистемы блоками

4.7.5.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/events/

4.7.5.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/events/

4.7.5.3 Пример ответа:

XML:

```
<events>
    <event>
        <description>Запись выключена</description>
        <id>{E56B09A0-1A50-E211-840E-005056C00008}</id>
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:43:27+04:00</ts>
    </event>
    <event>
        <description>Запись выключена</description>
        <id>{4482F63F-1A50-E211-840E-005056C00008}</id>
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:40:50+04:00</ts>
    </event>
    <event>
        <description>Запись выключена</description>
        <id>{35D4BE3E-1750-E211-840E-005056C00008}</id>
        <objectId>CAM:1</objectId>
        <ts>2012-12-27T15:19:16+04:00</ts>
    </event>
</events>
```

JSON:

```
[ {
  "id" : "{E56B09A0-1A50-E211-840E-005056C00008}",
  "description" : "Запись выключена",
  "ts" : "2012-12-27T15:43:27.000+04:00",
  "objectId" : "CAM:1"
}, {
  "id" : "{4482F63F-1A50-E211-840E-005056C00008}",
  "description" : "Запись выключена",
  "ts" : "2012-12-27T15:40:50.000+04:00",
  "objectId" : "CAM:1"
}, {
  "id" : "{35D4BE3E-1750-E211-840E-005056C00008}",
  "description" : "Запись выключена",
  "ts" : "2012-12-27T15:19:16.000+04:00",
  "objectId" : "CAM:1"
} ]
```

4.7.5.4 Параметры ответа:

Параметр	Описание
from	Самая старая дата промежутка поиска сообщений. (2012-12-27T15%3A19%3A16.000%2B04%3A00)

Параметр	Описание
to	Самая последняя дата промежутка поиска сообщений. (2012-12-27T15%3A19%3A16.000%2B04%3A00)
count	Максимальное количество сообщений в ответе [1, 200]. По-умолчанию 20. Сервер может вернуть чуть больше, если сообщений в базе данных осталось мало.
objectId	id объекта (CAM:1, GRAY:5 и т.д). Если параметр не задан, то возвращаются события всех объектов.

Коды возврата:

200 - OK
400 - неверный параметр (формат даты, например)
500 - ошибка
503 - ошибка соединения с ядром
504 - таймаут (ядро не вернуло данные в течение 2000 миллисекунд)

4.8 Отсылка команд на сервер

4.8.1 Общий формат запроса:

PUT http://IP-адрес:порт/web2/secure/configuration/{objectClass}:{id}/state/actions/{CMD}/execute

4.8.2 Параметры запроса:

Параметр	Обязательный	Описание
objectClass	Да	Название класса объектов
id	Да	Идентификатор объекта
CMD	Да	Команда *Внимание! Название команды должно быть указано в верхнем регистре.*

4.8.3 Пример запроса:

PUT http://127.0.0.1:8085/web2/secure/configuration/GRAY:2/state/actions/DISARM/execute

4.9 Макрокоманды

Макрокоманды (макросы) – это некоторая предопределённая последовательность реакций на определённые события. Макрокоманды создаются на сервере и имеют ID и название. Они похожи на действия с объектами, но не привязаны к объекту.

4.9.1 Получение списка макрокоманд

4.9.1.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/actions/

4.9.1.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/actions/

4.9.1.3 Пример ответа:

```
<actions>
    <action>
        <description>Начать запись по всем камерам</description>
        <id>2</id>
    </action>
    <action>
        <description>Снять с охраны все зоны</description>
        <id>1</id>
    </action>
</actions>
```

4.9.1.4 Параметры ответа:

Параметр	Описание
description	Название макрокоманды
id	Идентификатор макрокоманды

4.9.2 Получение параметров макрокоманд

4.9.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/actions/{id}/

4.9.2.2 Параметры запроса:

Параметр	Обязательный	Описание
id	Да	Идентификатор макрокоманды

4.9.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/actions/2/

4.9.2.4 Пример ответа:

```
<action>
    <description>Start recording by all cameras</description>
    <id>2</id>
</action>
```

4.9.2.5 Параметры ответа:

Параметр	Описание
description	Название макрокоманды

Параметр	Описание
id	Идентификатор макрокоманды

4.9.3 Запрос на выполнение макрокоманды на сервере

4.9.3.1 Общий формат запроса:

PUT http://IP-адрес:порт/web2/secure/configuration/{id}/state/actions/RUN/execute

4.9.3.2 Параметры запроса:

Параметр	Обязательный	Описание
id	Да	Тип и идентификатор макрокоманды в формате ТИП:ID (например MACRO:1)

4.9.3.3 Пример запроса:

PUT http://127.0.0.1:8085/web2/secure/configuration/MACRO:1/state/actions/RUN/execute

4.10 Видео

4.10.1 Запрос миниатюр (скриншотов)

4.10.1.1 Общий формат запроса:

4.10.1.1.1 1-й способ

GET http://IP-адрес:порт/web2/secure/video/image.jpg?cam.id={cam.id}&width={width}&height={height}&version={version}&login={login}&password={password}

4.10.1.2 Параметры запроса:

Параметр	Обязательный	Описание
cam.id	Да	Идентификатор камеры
width	Нет	Ширина кадра. Значение может быть в диапазоне [64, 1600]. Сервер автоматически округляет ширину до большего значения, кратного 4 Если нет, то размер возвращаемого изображения берётся из видеопотока
height	Нет	Высота кадра. Значение может быть в диапазоне [30, 1200] Если нет, то размер возвращаемого изображения берётся из видеопотока.
version	Нет	См. Версия продукта
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.10.1.2.1 2-й способ

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&command=frame.video&video_in={video_in}&imageWidth={imageWidth}&imageHeight={imageHeight}&login={login}&password={password}

4.10.1.3 Параметры запроса:

Параметр	Обязательный	Описание
version	Нет	См. Версия продукта
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
imageWidth	Нет	Ширина кадра. Значение может быть в диапазоне [64, 1600]. Сервер автоматически округляет ширину до большего значения, кратного 4 Если нет, то размер возвращаемого изображения берётся из видеопотока
imageHeight	Нет	Высота кадра. Значение может быть в диапазоне [30, 1200] Если нет, то размер возвращаемого изображения берётся из видеопотока.
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.10.1.4 Пример запроса:

4.10.1.4.1 1-й способ

GET http://127.0.0.1:8085/web2/secure/video/image.jpg?cam.id=5&width=85&version=4.7.8.0&login=USER&password=PASS

4.10.1.4.2 2-й способ

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&command=frame.video&video_in=CAM:1&imageWidth=400

4.10.1.5 Пример ответа:

В ответ придет jpeg изображение приблизительно запрошенного размера, либо код об ошибке и body нулевой длины (т.е. придут только заголовки).

Если произошла ошибка, то возвращается http код ошибки:
404 – камера отключена или не используется (disabled);
403 – неверный пароль;
426 – старая версия клиента;
429 – слишком много запросов;
444 – потерян сигнал по камере или камера отключена (коаксиальный провод отключен от платы);
503 – ошибка архива.

4.10.2 Запрос конфигурации

4.10.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/config.properties?version={version}&login={login}&password={password}

4.10.2.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.10.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/config.properties?version=4.7.8.0&login=USER&password=PASS

4.10.2.4 Пример ответа:

Текстовый файл config.properties.

Если пароль установлен, но не указан в запросе:

```
password.enabled=true
login.enabled=true
password.invalid=true#
```

Если пароль правильный или доступ разрешен без пароля:

```
password.enabled=true
login.enabled=true
password.invalid=false
cam.0.id=2
cam.0.name=Face
cam.0.rights=11
cam.1.id=3
cam.1.name=Camera 3
cam.1.rights=11
cam.2.id=5
cam.2.name=Camera 5
cam.2.rights=11
cam.2.telemetry_id=1.1
cam.count=3#
```

4.10.2.5 Параметры ответа:

Параметр	Описание
password.enabled	false - пароль не требуется true - нужен пароль
login.enabled	false - логин не требуется true - нужен логин
password.invalid	false - введён верный пароль true - введён неверный пароль

Параметр	Описание
cam.count	Общее количество камер в присланной конфигурации (id начинается с нуля)
cam.N.id	id камеры
cam.N.name	Название камеры
cam.N.rights	Права (они проверяются на сервере, но чтобы не показывать пользователю лишних опций, доступны и на клиенте). Параметр представляет собой флаги. Если флаг проставлен, то элемент интерфейса следует показывать, если нет, то скрывать. static final int RIGHT_VIEW = 0x1; // доступен просмотр живого видео (этот всегда проставлен в 1) static final int RIGHT_CONTROL = 0x2; // управление (телеметрия, постановка и снятие с охраны) static final int RIGHT_CONFIG = 0x4; // зарезервирован static final int RIGHT_HISTORY = 0x8; // доступ к архиву
cam.N.telemetry_id	id телеметрии (может отсутствовать, если телеметрии нет, тогда необходимо скрывать элементы управления телеметрией)

4.10.3 Запрос видео

4.10.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&video_in={video_in}&normalize={normalize}&imageWidth={imageWidth}&imageHeight={imageHeight}&fps={fps}&login={login}&password={password}

4.10.3.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта. Если указать в запросе версию 4.10.0.0, в результате будет получен поток в формате MJPEG без xml-вставок, который можно отображать на web-странице в браузерах Chrome и FireFox при помощи тэга IMG. Данная функция реализована как для живого, так и для архивного видео. *Примечание. Допускается одновременное получение не более 6 потоков видео.*
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
sessionid	Нет	Идентификатор сессии
imageWidth	Нет	Ширина кадра. Значение может быть в диапазоне [64, 1600]. Сервер автоматически округляет ширину до большего значения, кратного 4
imageHeight	Нет	Высота кадра. Значение может быть в диапазоне [30, 1200]. Если нет, то размер возвращаемого изображения берётся из видеопотока
normalize	Нет	true - растягивает изображение, если кадр приходит в некорректных пропорциях
fps	Нет	Частота кадров видео
login	Нет	Имя пользователя ПК *Интеллект*, если установлен

Параметр	Обязательный	Описание
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.10.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&sessionid=1234567890&video_in=CAM:1&imageWidth=200&fps=1&login=USER&password=PASS

4.10.3.4 Пример ответа:

```
<html>
 <head/>
 <body>
  <img src="путь к потоку mjpeg"/>
 </body>
</html>
```

4.10.4 Curl запрос видео. Формат основного потока

4.10.4.1 Общий формат запроса:

curl -v "http://IP-адрес:порт/web2/secure/video/action.do?fps={fps}&imageHeight={imageHeight}&login={login}&normalize={normalize}&password={password}&sessionid={sessionid}&version={version}&video_in={video_in}" --output ~/{output file}.bin

4.10.4.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта. Если указать в запросе версию 4.10.0.0, в результате будет получен поток в формате MJPEG без xml-вставок, который можно отображать на web-странице в браузерах Chrome и FireFox при помощи тэга IMG. Данная функция реализована как для живого, так и для архивного видео. *Примечание. Допускается одновременное получение не более 6 потоков видео.*
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
sessionid	Нет	Идентификатор сессии
imageWidth	Нет	Ширина кадра. Значение может быть в диапазоне [64, 1600]. Сервер автоматически округляет ширину до большего значения, кратного 4. Если нет, то размер возвращаемого изображения берётся из видеопотока
imageHeight	Нет	Высота кадра. Значение может быть в диапазоне [30, 1200]. Если нет, то размер возвращаемого изображения берётся из видеопотока
fps	Нет	Частота кадров видео

Параметр	Обязательный	Описание
normalize	Нет	true - растягивает изображение, если кадр приходит в некорректных пропорциях
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен
output file name	Да	Имя файла, в который будет осуществляться вывод запрашиваемого видео

4.10.4.3 Пример запроса:

curl -v "http://127.0.0.1:8085/web2/secure/video/action.do?fps=1&imageHeight=360&login=1&normalize=true&password=1&sessionid=A4D98DDE-A535-49E4-9FB5-FAD441CBBA43&version=4.10.0.0&video_in=CAM:1" --output ~/output.bin

4.10.4.4 Пример ответа:

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0* Connected to
172.19.2.6 (172.19.2.6) port 8085 (#0)
* Server auth using Basic with user '1'
> GET /web2/secure/video/action.do?
fps=1&imageHeight=360&login=1&normalize=true&password=1&sessionid=A4D98DDE-A535-49E4-9FB5-
FAD441CBBA43&version=4.10.0.0&video_in=CAM:1 HTTP/1.1
> Host: 172.19.2.6:8085
> Authorization: Basic MTox
> User-Agent: curl/7.74.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Set-Cookie: JSESSIONID=ae532304-e6dc-4a19-8c4f-1e10656140f2; Path=/web2; HttpOnly
< Set-Cookie: rememberMe=deleteMe; Path=/web2; Max-Age=0; Expires=Sun, 24-Jan-2021 08:09:52 GMT
< Server: ITV-Intellect-Webserver/4.11.2.3000
< Date: Mon, 25 Jan 2021 08:09:52 GMT
< Cache-Control: no-store, no-cache, must-revalidate, max-age=0
< Connection: close
< Content-Type: multipart/x-mixed-replace;boundary=videoframe
< Pragma: no-cache
< X-CameraID: 1
< X-SessionID: A4D98DDE-A535-49E4-9FB5-FAD441CBBA43
< Closing connection 0
<
{ [32768 bytes data]
```

4.10.4.5 Параметры ответа:

В заголовке Content-Type указываются следующие параметры:

Параметр	Описание
multipart/x-mixed-replace	Указывает, что каждая следующая часть контента должна заменять собой предыдущую
boundary	Указывает текстовый разделитель между частями контента

4.10.4.6 Пример ответа:

В файле, в который осуществлялся вывод, для каждой части запрашиваемого контента будут содержаться следующие параметры:

```
--videoframe
Content-Type: image/jpeg
Content-Length: 93081
X-Width: 480
X-Height: 360
X-Timestamp: 0.000
X-Time: 2021-01-25T10:06:42.816+03:00
```

4.10.4.7 Параметры ответа:

Параметр	Описание
Content-Type	Тип контента
Content-Length	Размер части контента
X-Width	Ширина изображения
X-Height	Высота изображения
X-Time	Абсолютное время формирования фрейма
X-Timestamp	Относительное время фрейма в секундах (относительно начала потока)

4.10.4.8 Пример ответа:

В случае завершения потока по вине сервера может прийти завершающий пакет:

```
--videoframe
Content-Type: image/jpeg
Content-Length: 106
<video_in>
  <sessionid>A4D98DDE-A535-49E4-9FB5-FAD441CBBA43</sessionid>
  <video_in>CAM:1</video_in>
  <newstate>closed</newstate>
  <errcode>103</errcode>
</video_in>
```

4.10.4.9 Параметры ответа:

Параметр	Описание
sessionid	id сессии (тот же что и при старте)
video_in	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
errcode	Код ошибки: • 100 – отсутствие ошибки. • 101 – слишком много подключенных пользователей. • 102 – неверный пароль (пароль, теоретически, могут поменять в любой момент работы). • 103 – видео недоступно. • 104 – старая версия клиента. Обновите версию.

4.10.5 Управление камерой

4.10.5.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&cam.id={cam.id}
&target=CAM&targetid={targetid}&command={command}&login={login}&password={password}

4.10.5.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
cam.id	Да	Идентификатор камеры
sessionid	Нет	Идентификатор сессии
target	Да	CAM - идентификатор класса объекта "Камера"
targetid	Да	Соответствует cam.id
command	Да	Команды управления камерой: • Начало записи: REC • Остановка записи: REC_STOP • Постановка на охрану: ARM • Снятие с охраны: DISARM
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.10.5.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?
version=4.10.0.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=REC

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=REC_STOP

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=ARM

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&sessionid=29101F1&cam.id=1&target=CAM&targetid=1&command=DISARM

4.10.6 Получение авторизованной ссылки на камеру по token ключу

Для получения заранее авторизованной ссылки на камеру (для получения как живого или архивного видео) необходимо:

- Выполнить запрос для получения токена:
- Выполнить запрос по полученному токену.

4.10.6.1 Общий формат запроса для получения токена:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&video_in={video_in}&enable_token_auth={enable_token_auth}&valid_token_hours={valid_token_hours}&login={login}&password={password}

4.10.6.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
sessionid	Нет	Идентификатор сессии
targetid	Да	Соответствует cam.id
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен
enable_token_auth	Да	Для включения авторизации по токену указать значение "**1**"
valid_token_hours	Нет	Время действия токена в часах. Максимальное значение - неделя. Токен перестает действовать через указанный период времени Значение по-умолчанию 12 часов

4.10.6.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&sessionid=FC126734&video_in=CAM:1&enable_token_auth=1&valid_token_hours=1&login=USER&password=PASS

4.10.6.4 Пример ответа:

```
{
"path" : "action.do?hmac=GAqUa429sjY2E9jCTpuYeaMqReW3Y7HI"
}
```

4.10.6.5 Параметры ответа:

Параметр	Описание
hmac	Токен

4.10.6.6 Общий формат запроса по полученному токену через Веб-сервер 2.0:

GET http://IP-адрес:порт/web2/secure/video/action.do?hmac={hmac}

4.10.6.7 Параметры запроса:

Параметр	Обязательный	Описание
hmac	Да	Полученный ранее токен

4.10.6.8 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?hmac=GAqUa429sjY2E9jCTpuYeaMqReW3Y7HI

4.10.6.9 Общий формат запроса по полученному токену через **Web сервер 1**:

> **Примечание**
>
> Данный способ запроса является устаревшим. Рекомендуется выполнять запрос через модуль **Веб-сервер 2.0**.

GET http://IP-адрес:{port}/action.do?hmac={hmac}

4.10.6.10 Параметры запроса:

Параметр	Обязательный	Описание
port	Да	Номер порта, указанный на панели настройки объекта **Web сервер** для подключения к HTTP-серверу (см. Задание параметров подключения Клиентов к Web серверу)
hmac	Да	Полученный ранее токен

4.10.6.11 Пример запроса:

GET http://127.0.0.1:80/action.do?hmac=GAqUa429sjY2E9jCTpuYeaMqReW3Y7HI

4.11 Управление телеметрией

4.11.1 Общий формат запроса:

GET http://{login}:{password}@IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={session_id}
&cam.id={cam_id}&target=PTZ&targetid={PTZ_device_id}&command={command}&login={login}&password={password}
&speed={speed}&preset={preset}

Внимание!

Если логин и пароль установлен, то в запросе их необходимо указывать дважды: перед IP-адресом и в виде параметров. В обоих случаях это один и тот же логин и пароль пользователя ПК *Интеллект*.

4.11.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
session_id	Да	Идентификатор сессии
cam_id	Да	Идентификатор камеры
PTZ_device_id	Да	Идентификатор устройства управления телеметрией, связанного с камерой (можно получить при запросе конфигурации – см. Получение списка всех объектов сервера).
command	Да	Выполняемая команда. Может принимать следующие значения: • RIGHT – поворот направо. • UP – перемещение вверх. • LEFT – поворот налево. • DOWN – перемещение вниз. • ZOOM_IN – увеличение масштаба. • ZOOM_OUT – уменьшение масштаба. • GO_PRESET – перейти в определенный пресет. • POINTMOVE – зуммирование выделенной точки на изображении (x, y). • AREAZOOM – зуммирование выделенной области изображения (x,y,w,h).
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен
speed	Да	Скорость отработки команды (от 0 до 10). При управлении по сети из-за задержек лучше использовать низкие значения.
preset	Нет	Номер пресета (число). Обязательный параметр только для command=GO_PRESET. В остальных случаях его значение игнорируется. **x** – координата x относительно размера изображения. Может принимать значения от 0.0 до 1.0. Обязательный параметр только для command=POINTMOVE или command=AREAZOOM. В остальных случаях его значение игнорируется. **y** – координата y относительно размера изображения. Может принимать значения от 0.0 до 1.0. Обязательный параметр только для command=POINTMOVE или command=AREAZOOM. В остальных случаях его значение игнорируется. **w** – ширина области зуммирования y относительно размера изображения. Может принимать значения от 0.0 до 1.0. Обязательный параметр только для command=AREAZOOM. В остальных случаях его значение игнорируется. **h** – высота области зуммирования относительно размера изображения. Может принимать значения от 0.0 до 1.0. Обязательный параметр только для command=AREAZOOM. В остальных случаях его значение игнорируется.

4.11.3 Пример запроса:

GET http://user:pass@127.0.0.1:8085/web2/secure/video/action.do?version=4.10.0.0&cam.id=5&target=PTZ&targetid=1.1&command=RIGHT&login=USER&password=PASS&speed=2

4.12 Работа с архивом

Поток из видеоархива присылается в таком же формате, что и живое видео.

4.12.1 Получение списка записей (1-й способ)

Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&video_in={video_in}
&command=arc.intervals&time_from={time_from}&time_to={time_to}&max_count={max_count}
&split_threshold={split_threshold}&login={login}&password={password}

4.12.1.1 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	Нет	Идентификатор сессии
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
command	Да	Команда для получения списка записей: arc.intervals
time_from	Да	Начало интересуемого диапазона времени
time_to	Нет	Конец интересуемого диапазона времени
max_count	Нет	Максимальное количество записей в ответе
split_threshold	Нет	Время для объединения нескольких интервалов (в секундах). Интервалы, расстояние между которыми будет меньше заданного, будут объединены в один
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен
format	Нет	Задает формат ответа (см. Общие сведения о HTTP API)

4.12.1.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:
5&command=arc.intervals&time_from=2013-03-20T00:00:00.000+04:00&time_to=2013-03-22T23:59:59.999+04:00&max_count=1
00&split_threshold=10399&login=USER&password=PASS

4.12.1.3 Пример ответа:

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<records count="1" complete="YES" sort="INCREASE">
    <record>
        <from>2011-09-01T00:00:00-05:00</from>
        <to>2011-09-01T00:00:35-05:00</to>
    </record>
    <record>
        <from>2011-09-01T00:00:35-05:00</from>
        <to>2011-09-01T00:01:10-05:00</to>
    </record>
</records>
```

JSON:

```
{ 'count' : 1,
  'complete' : 'YES',
  'sort' : 'INCREASE',
  'cam' : '1',
  'records' : [ {
     'from' : '2019-01-22T12:41:10.144+03:00',
    'to' : '2019-01-23T08:28:47.346+03:00'
  }]
}
```

4.12.2 Получение списка записей (2-й способ)

4.12.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/archive/{CAM:id}/{DATE}/?[splitThreshold={splitThreshold}]&[days={days}]

4.12.2.2 Параметры запроса:

Параметр	Обязательный	Описание
CAM:id	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
DATE	Да	Дата начала получения архива. Дата интерпретируется как локальное для сервера
splitThreshold	Да	Если разница между окончанием предыдущей записи и началом следующей меньше этого числа (в миллисекундах), то записи объединяются в одну. Чтобы никакие записи не объединялись, нужно указать splitTreshold=0. [default: 50]
days	Да	Количество дней, от текущего, за которые требуется получить архив. [default: 1]

4.12.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/archive/CAM:2/2011-12-30/?[splitThreshold=50]&[days=1]

GET http://127.0.0.1:8085/web2/secure/archive/CAM:1/2013-10-18/?splitTreshold=2000 – получить записи за 18 ноября 2013 года и слепить все записи, промежуток между которыми меньше 2000 миллисекунд.

GET http://127.0.0.1:8085/web2/secure/archive/CAM:1/2013-10-18/?days=10 – получить записи за 10 дней, начиная с 18 ноября 2013 года.

4.12.2.4 Пример ответа:

XML:

```
<?xml version="1.0" encoding="UTF-16"?>
<days>
    <day>
        <id>2013-11-10T00:00:00-02:00</id>
        <records>
            <from>2013-11-10T18:44:01.579-02:00</from>
            <to>2013-11-10T18:44:09.717-02:00</to>
        </records>
    </day>
    <day>
        <id>2013-11-18T00:00:00-02:00</id>
        <records>
            <from>2013-11-18T18:38:30.252-02:00</from>
            <to>2013-11-18T18:38:56.942-02:00</to>
        </records>
        <records>
            <from>2013-11-18T18:39:08.321-02:00</from>
            <to>2013-11-18T18:39:10.080-02:00</to>
        </records>
    </day>
</days>
```

JSON:

```
[ {
  "id" : "2013-11-10T00:00:00.000-02:00",
  "records" : [ {
    "from" : "2013-11-10T18:44:01.579-02:00",
    "to" : "2013-11-10T18:44:09.717-02:00"
  } ]
}, {
  "id" : "2013-11-18T00:00:00.000-02:00",
  "records" : [ {
    "from" : "2013-11-18T18:38:30.252-02:00",
    "to" : "2013-11-18T18:38:56.942-02:00"
  }, {
    "from" : "2013-11-18T18:39:08.321-02:00",
    "to" : "2013-11-18T18:39:10.080-02:00"
  } ]
} ]
```

Получение записей за месяц (показывает, в какие дни сентября есть записи):

GET http://127.0.0.1:8085/web2/secure/archive/CAM:2/2011-12/

XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<days>
    <day>
        <id>2011-09-02T00:00:00-05:00</id>
    </day>
    <day>
        <id>2011-09-03T00:00:00-05:00</id>
    </day>
    <day>
        <id>2011-09-05T00:00:00-05:00</id>
    </day>
</days>
```

JSON:

```
[ {
  "id" : "2011-09-01T00:00:00-0500",
  "records" : [ ]
}, {
  "id" : "2011-09-03T00:00:00-0500",
  "records" : [ ]
}, {
  "id" : "2011-09-01T00:00:00-0500",
  "records" : [ ]
} ]
```

Если записей нет, то присылается:

XML:

```
[<days/>
JSON:
[]
```

4.12.3 Получение видео из архива - "arc.play"

4.12.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&video_in={video_in}
&command=arc.play&time_from={time_from}&time_to={time_to}&login={login}&password={password}
&speed_factor={speed_factor}

4.12.3.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	Нет	Идентификатор сессии
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"

Параметр	Обязательный	Описание
command	Да	Команда для получения видео из архива: arc.play
time_from	Да	Время начала проигрывания архива
time_to	Нет	Время завершения проигрывания архива (если не указано, будет отдан весь архив до последней записи)
imageWidth	Нет	Ширина в пикселях (если не указано или 0, рассчитывается автоматически с сохранением пропорций)
imageHeight	Нет	Высота в пикселях (если не указано или 0, рассчитывается автоматически с сохранением пропорций)
fps	Нет	Максимальная частота кадров в секунду (если не указано или 0, часта кадров не будет ограничиваться)
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен
speed_factor	Нет	Задает скорость воспроизведения. Параметр может принимать любое целое или дробное значение не меньше 0. Примеры значений: 0 – воспроизведение с максимально возможной скоростью (зависит от пропускной способности сети и нагрузки на диск) 1 – воспроизведение с нормальной скоростью x1 (по умолчанию) 0.1 – замедленное воспроизведение со скоростью x1/10 2 – ускоренное воспроизведение со скоростью x2
format	Нет	Задает формат ответа (см. Общие сведения о HTTP API)

4.12.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:5&command=arc.play&time_from=2013-03-22T13:04:52.312+04:00&time_to=2013-03-22T13:16:31.873+04:00&login=USER&password=PASS&speed_factor=1

4.12.3.4 Пример ответа:

При завершении потока придет завершающий пакет с newstate=closed и errcode=100.

4.12.4 Получение одного кадра из архива - "arc.frame"

4.12.4.1 Общий формат запроса (1-й способ):

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&video_in={video_in}&command=arc.frame&time={time}&range={range}&login={login}&password={password}

4.12.4.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	Нет	Идентификатор сессии
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
command	Да	Команда для получения одного фрейма: arc.frame
time	Да	Время кадра
range	Нет	Время в секундах, для задания диапазона поиска ближайшего фрейма относительно time (если не указан, ищется ближайший по всему архиву)
imageWidth	Нет	Ширина в пикселях (если не указано или 0, рассчитывается автоматически с сохранением пропорций)
imageHeight	Нет	Высота в пикселях (если не указано или 0, рассчитывается автоматически с сохранением пропорций)
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.12.4.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&video_in=CAM:5&command=arc.frame&time=2013-03-22T13:04:52.312+04:00&range=0.1&login=USER&password=PASS

4.12.4.4 Общий формат запроса (2-й способ):

GET http://IP-адрес:порт/action.do?version={version}&video_in={video_in}&command=arc.frame&time={time}

4.12.4.5 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
video_in	Да	Идентификатор камеры в формате "ТИП:ИДЕНТИФИКАТОР", например, "CAM:1"
command	Да	Команда для получения одного фрейма: arc.frame
time	Да	Время кадра

4.12.4.6 Пример запроса:

GET http://127.0.0.1:8085/action.do?version=4.9.0.0&video_in=CAM:1&command=arc.frame&time=2018-08-12T22:29:06Z

4.12.4.7 Пример ответа:

В обоих случаях в ответ придут http-заголовки и ближайший фрейм из диапазона [time - range, time + range] в формате jpeg. Если фрейма в диапазоне не будет тело в ответе будет пустым.

4.13 Нотификация

Приложение при соединении с сервисом может осуществить подписку на сообщения APNS. В этом случае при выходе из программы на устройство будут приходить уведомления о тех или иных событиях.

Используются системы нотификации APNS(iOS), C2DN (Android) и т.д.

4.13.1 Формат сообщения APN

```
{
"aps" : {
                    "alert" : "Motion Detected",
                    "badge" : 2 //порядковый номер сообщения. Номера выдаются по порядку после
момента последней подписки.
            },
     "e" : {
                "srv" : "XXX",  //id сервера. Уникальный в рамках одного устройства iOS
                "stt" : 88,   //id состояния (см. Список состояний для определённого класса
объектов)
                "obj" : "6", //id объекта
                "ts" : "2010-08-02T23:30:00Z" //время отсылки события
            }
}
```

4.13.2 Подписка на сообщения APNS

4.13.2.1 Общий формат запроса:

POST http://IP-адрес:порт/web2/secure/subscription/

Тело POST должно содержать информацию о создаваемой подписке. Принимается только формат JSON. Требуется корректно проставлять заголовок Content-Type.

```
JSON
Content-Type  :  application/json
{
    "userparam" : "{userparam}",
    "deviceid" : "{deviceid}"
}
```

4.13.2.2 Параметры запроса:

Параметр	Обязательный	Описание
deviceid	Да	device token (APNs), registration id (в случае C2DN) и т.д Не должно быть пустым, должно быть длиной от 5 до 150 символов и содержать только цифры и буквы английского алфавита
userparam	Нет	Логин пользователя. Может быть пустой

4.13.2.3 Пример запроса:

POST http://127.0.0.1:8085/web2/secure/subscription/

```
JSON
Content-Type  :  application/json
{
    "userparam" : "John doe",
    "deviceid" : "FC126734454FGD5"
}
```

4.13.2.4 Пример ответа:

Ответ с кодом “201 Created” означает, что подписка прошла успешно.

Ответ с кодом 400 означает, что параметры заданы не верно.

4.13.3 Аннулирование подписки на сообщения APNS

Аннулирование подписки происходит в следующих случаях:

- Пользователь подписался на события с другого устройства;
- Сменился device token или registration id;
- Другой пользователь подписался на события с данного устройства;
- Произошла ручная отписка от сообщений.

4.13.3.1 Общий формат запроса:

DELETE http://IP-адрес:порт/web2/web2/secure/subscription/[{deviceId}

4.13.3.2 Параметры запроса:

Параметр	Обязательный	Описание
deviceid	Да	device token (APNs), registration id (в случае C2DN) и т.д

4.13.3.3 Пример запроса:

DELETE http://127.0.0.1:8085/web2/secure/subscription/[FC126734454FGD5]

4.13.3.4 Пример ответа:

Ответ с кодом “204 No Content” означает, что аннулирование подписки прошло успешно.

4.14 Звук

4.14.1 Получение живого звука

4.14.1.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}
&command=audio.play&audio_in={audio_in}&format={format}&login={login}&password={password}

4.14.1.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	No	Идентификатор сессии
command	Да	Команда получения живого звука: **audio.play**
format	Да	Формат аудиоданных
audio_in	Да	Идентификатор микрофона в формате "ТИП:ИДЕНТИФИКАТОР", например, "MIC:1"
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.14.1.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?
version=4.9.0.0&sessionid=FC126734&command=audio.play&audio_in=MIC:5&format=L16&login=USER&password=PASS

4.14.1.4 Пример ответа:

В ответ будут получены аудиопакеты в следующем виде:

```
HTTP/1.0 200 OK
Connection: close
Server: ITV-Intellect-Webserver/4.9.0.0
Cache-Control: no-store,no-cache,must-revalidate,max-age=0
Pragma: no-cache
Date: Mon, 13 Jan 2013 10:44:27 GMT
Content-Type: multipart/mixed;boundary=audioframe

--audioframe
Content-Type: text/xml
Content-Length: 138

<audio_in>
  <sessionid>FC126734</sessionid>
  <audio_in>MIC:5</audio_in>
  <newstate>started</newstate>
  <errcode>100</errcode>
</audio_in>
--audioframe
Content-Type: audio/L16;rate=8000;channels=1
Content-Length: 1024
X-Time: 2013-03-22T13:16:31.371+04:00

<audio packet PCM16>
--audioframe
Content-Type: audio/L16;rate=8000;channels=1
Content-Length: 1278
X-Time: 2013-03-22T13:16:31.873+04:00
<audio packet PCM16>
```

4.14.2 Остановка получения живого звука

4.14.2.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}
&command=audio.stop&audio_in={audio_in}&login={login}&password={password}

4.14.2.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	No	Идентификатор сессии
command	Да	Команда остановки получение живого звука: **audio.stop**
audio_in	Да	Идентификатор микрофона в формате "ТИП:ИДЕНТИФИКАТОР", например, "MIC:1"
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.14.2.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?
version=4.9.0.0&sessionid=29101F1&command=audio.stop&audio_in=MIC:5&login=USER&password=PASS

4.14.2.4 Пример ответа:

В этом случае в потоке придёт завершающий xml пакет:

```
--audioframe
Content-Type: text/xml
Content-Length: 106
<audio_in>
  <sessionid>FC126734</sessionid>
  <audio_in>MIC:5</audio_in>
  <newstate>closed</newstate>
  <errcode>100</errcode>
 </audio_in>
```

4.14.3 Проигрывание звука из архива

4.14.3.1 Общий формат запроса:

GET http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}
&command=arc.play&audio_in={audio_in}&format={format}&time_from={time_from}&time_to={time_to}&login={login}
&password={password}

4.14.3.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	Нет	Идентификатор сессии
command	Да	Команда получения звука из архива: **arc.play**
format	Да	Формат аудиоданных
audio_in	Да	Идентификатор микрофона в формате "ТИП:ИДЕНТИФИКАТОР", например, "MIC:1"
time_from	Да	Время начала проигрывания архива
time_to	Да	Время завершения проигрывания архива
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.14.3.3 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&sessionid=29101F1&command=arc.play&audio_in=MIC:5&format=L16&time_from=2013-03-22T13:16:31.873+04:00&time_to=2013-03-22T13:04:52.312+04:00&login=USER&password=PASS

4.14.3.4 Пример ответа:

Поток приходит в том же виде, что и при живом звуке (см. Получение живого звука). При завершении данных приходит завершающий xml пакет (см. Остановка получения живого звука).

4.14.4 Отправка живого звука

4.14.4.1 Общий формат запроса:

POST http://IP-адрес:порт/web2/secure/video/action.do?version={version}&sessionid={sessionid}&command=audio.receive&audio_out={audio_out}&login={login}&password={password}

4.14.4.2 Параметры запроса:

Параметр	Обязательный	Описание
version	Да	См. Версия продукта
sessionid	Да	Идентификатор сессии
command	Да	Команда отправки живого звука: **audio.receive**
audio_out	Да	Идентификатор спикера в формате "ТИП:ИДЕНТИФИКАТОР", например, "SPEAKER:1"
login	Нет	Имя пользователя ПК *Интеллект*, если установлен
password	Нет	Пароль пользователя ПК *Интеллект*, если установлен

4.14.4.3 Пример запроса:

POST http://127.0.0.1:8085/web2/secure/video/action.do?version=4.9.0.0&sessionid=FC126734&command=audio.receive&audio_out=SPEAKER:3&login=USER&password=PASS

4.14.4.4 Пример ответа:

Отправка звука идёт последовательной передачей пакетов командами:

```
Content-type: audio/L16;rate=8000;channels=1
Connection: keep-alive
```

Далее происходит передача аудиопакета.

4.14.4.5 Параметр ответа:

Параметр	Описание
L16	Формат звука только L16

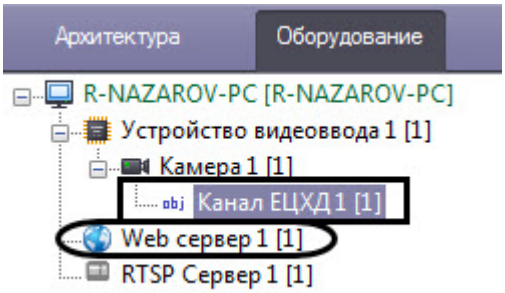
Параметр	Описание
rate	Любое разумное значение
channels	Канал от 1 до 6

4.15 Команды, используемые для интеграции ЕЦХД

ЕЦХД – государственная информационная система "Единый центр хранения и обработки данных".

В данном разделе описаны специальные http-запросы, используемые для интеграции ПК *Интеллект* с ЕЦХД. Для их работы необходимо, чтобы возможность использования таких запросов была включена на этапе настройки программы – см. Выбор rtsp сервера для обработки запросов ЕЦХД.

Запросы отправляются на **Web сервер** и выполняются только для тех камер, на базе которых создан объект **Канал ЕЦХД**.



4.15.1 Получение информации об устройстве

4.15.1.1 Общий формат запроса:

GET http://IP-адрес:порт/getdeviceinfo

4.15.1.2 Пример запроса:

GET http://127.0.0.1:80/getdeviceinfo

4.15.1.3 Пример ответа:

```
{
        "firmware version": "1.2.3 Rev B.",
        "vendor": "Vendor Title Ltd"
        "model": "Device Model",
        "serial_number": "12345ABCDEF",
        "ptz-status": "not supported"
}
```

4.15.1.4 Параметры ответа:

Параметр	Описание
firmware version	Версия прошивки. Также можно добавить произвольный префикс с помощью ключа реестра AdditionalVersionString – см. Справочник ключей реестра.

Параметр	Описание
vendor	Производитель
model	Модель
serial_number	Серийный номер
ptz-status	Поддержка PTZ

4.15.2 Получение списка идентификаторов камер

4.15.2.1 Общий формат запроса:

GET http://IP-адрес:порт/getcameras

4.15.2.2 Пример запроса:

GET http://127.0.0.1:80/getcameras

4.15.2.3 Пример ответа:

```
{
        "cameras": [
                {
                        "id": 1,
                        "channel": 1,
                        "status": "working"
                },
                {
                        "id": 2,
                        "channel ": 2,
                        "status": "signaIJost"
                }
                        ]
}
```

4.15.2.4 Параметры ответа:

Параметр	Описание
id	Идентификатор камеры
channel	Канал камеры
status	Статус камеры

4.15.3 Диапазоны доступных архивных записей

4.15.3.1 Общий формат запроса:

GET http://IP-адрес:порт/getarchiveranges?cameraid={cameraid}

или

GET http://IP-адрес:порт/getavailablearchiveranges?cameraid={cameraid}

4.15.3.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры

4.15.3.3 Пример запроса:

GET http://127.0.0.1:80/getarchiveranges?cameraid=1

4.15.3.4 Пример ответа:

Возвращает периоды времени, за которое доступны архивные записи с указанного средства видеонаблюдения. По умолчанию фрагменты в ответе на запрос "склеиваются" (объединяются), если расстояние между ними составляет менее 5 секунд. Изменить данный период времени можно при помощи ключа реестра **SplitArchiveIntervals** (см. Справочник ключей реестра).

```
{
        "cameraid": 1,
        "ranges": [
                {
                        "from": 1412121600, //unixtime
                        "to": 1412172000
                },
                {
                        "from": 1412186400,
                        "to": 1412188200
                }
                        ]
}
```

4.15.4 Работа с видеопотоками

4.15.4.1 Запрос GetLiveUrl

GetLiveUrl - возвращает rtsp URL «живого» видеопотока для указанной камеры.

4.15.4.1.1 Общий формат запроса:

GET http://IP-адрес:порт/getliveurl?cameraid={cameraid}

4.15.4.1.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры

4.15.4.1.3 Пример запроса:

GET http://127.0.0.1:80/getliveurl?cameraid=1

4.15.4.1.4 Пример ответа:

```
{
"rtspurl": "rtsp://device-address/somelivemediastream0"
}
```

4.15.4.2 Запрос GetArclliveURL

GetArclliveURL - возвращает rtsp URL архивного видеопотока, отдаваемого регистратором, для указанной камеры, начиная с fromdatetime (и опционально, заканчивая todatetime).

4.15.4.2.1 Общий формат запроса:

GET http://IP-адрес:порт/getarchiveurl?cameraid={cameraid}&fromdatetime={fromdatetime}&todatetime={todatetime}

4.15.4.2.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры
fromdatetime	Да	Время начала фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС
todatetime	Нет	Время окончания фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС (если не указано, будет отдан весь архив до последней записи)

4.15.4.2.3 Пример запроса:

GET http://127.0.0.1:80/getarchiveurl?cameraid=1&fromdatetime=2014-10-01T00:00:00&todatetime=2014-10-01T01:20:05

4.15.4.2.4 Пример ответа:

```
{
"rtspurl": "rtsp://deviceladdress/somearchivemediastream?somedatetimetoken"
}
```

4.15.5 Выгрузка архивов

4.15.5.1 Общий формат запроса:

GET http://IP-адрес:порт/downloadarchivefile?cameraid={cameraid&fromdatetime={fromdatetime}&todatetime={todatetime}

4.15.5.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры
fromdatetime	Да	Время начала фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС

Параметр	Обязательный	Описание
todatetime	Да	Время окончания фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС

4.15.5.3 Пример запроса:

GET http://127.0.0.1:80/downloadarchivefile?cameraid=1&fromdatetime=2014-10-01T00:00:00&todatetime=2014-10-01T01:20:05

4.15.5.4 Пример ответа:

```
HTTP/l.1 200 OK
Content-Type: application/octet-stream
```

Также в результате выполнения команды будет получен файл с расширением es (например, Camera[4] (2019-08-13T11_00_00 - 2019-08-13T12_10_00).es). Чтобы воспроизвести этот файл, необходимо конвертировать его с помощью утилиты ffmpeg. Загрузить эту утилиту можно на официальном сайте https://ffmpeg.org/

Команда для конвертации файла в кодеке .264:

ffmpeg -i "C:\путь до файла в кодеке .264\Camera[5].es" -c:v copy -bsf:v h264_mp4toannexb -c:a copy -f avi output.avi

Команда для конвертации файла в кодеке .265:

ffmpeg -i "C:\путь до файла в кодеке .265\Camera[5].es" -c:v copy -bsf:v hevc_mp4toannexb -c:a copy -f avi output.avi

В результате выполнения команды будет создан файл в формате .avi (output.avi).

4.15.6 Экспорт архива

4.15.6.1 Создание задания на экспорт архива

По умолчанию экспорт архива осуществляется в формат mp4. Изменить формат можно при помощи ключа реестра ExportContainerFormat (см. Справочник ключей реестра). Поддерживается экспорт архива, записанного в формате H.264 или MPEG4.

4.15.6.1.1 Общий формат запроса:

POST http://IP-адрес:порт/createarchivetask

```
Content Type: application/json
Content:
{
 "CameraId": "{CameraId}",
 "From": "{From}",
 "To": "{To}"
}
```

4.15.6.1.2 Параметры запроса:

Параметр	Обязательный	Описание
CameraId	Да	Идентификатор камеры

Параметр	Обязательный	Описание
From	Да	Время начала фрагмента архива в формате UTC: ГГГГ-ММ-ДДТЧЧ:ММ:CCZ
To	Нет	Время окончания фрагмента архива в формате UTC: ГГГГ-ММ-ДДТЧЧ:ММ:CCZ (если не указано, будет отдан весь архив до последней записи)

4.15.6.1.3 Пример запроса:

POST http://127.0.0.1:80/createarchivetask

```
Content Type: application/json
Content:
{
 "CameraId": "1",
 "From": "2016-06-27T15:10:00.00Z",
 "To": "2016-06-27T15:20:00.00Z"
}
```

4.15.6.1.4 Пример ответа:

В папке экспорта (по умолчанию C:\Users\User\Documents\Intellect\export) создаётся папка с соответствующим идентификатором задачи (084b56a5-bd49-4327-82db-9bc911f7ff96) и mp4-файлом внутри.

```
{
    "CameraId" : "1",
    "From" : "2016-06-27T15:10:00.00Z",
    "To" : "2016-06-27T15:20:00.00Z",
    "ArchiveTaskId" : "084b56a5-bd49-4327-82db-9bc911f7ff96",
    "ErrorMessage" : null,
    "State" : "Created"
}
```

4.15.6.1.5 Параметры ответа:

Параметр	Описание
CameraId	Идентификатор камеры
From	Время начала фрагмента архива в формате UTC: ГГГГ-ММ-ДДТЧЧ:ММ:CCZ
To	Время окончания фрагмента архива в формате UTC: ГГГГ-ММ-ДДТЧЧ:ММ:CCZ
ArchiveTaskId	Идентификатор задачи
ArchiveTaskId	Сообщения об ошибках
State	Результат создания задачи

4.15.6.2 Получение статуса экспорта

4.15.6.2.1 Общий формат запроса:

GET http://IP-адрес:порт/getarchivetaskstatus?archivetaskid={archivetaskid}

4.15.6.2.2 Параметры запроса:

Параметр	Обязательный	Описание
archivetaskid	Да	Идентификатор задачи

4.15.6.2.3 Пример запроса:

GET http://127.0.0.1:80/getarchivetaskstatus?archivetaskid=104b38d4-07d7-4d2f-84da-49b3e255d2bf

4.15.6.2.4 Пример ответа:

```
{
    "Percents" : 100,
    "Url" : "http://192.168.15.182:80/download?file=104b38d4-07d7-4d2f-84da-49b3e255d2bf",
    "CameraId" : "1",
    "From" : "2016-06-27T15:10:00.000+03:00",
    "To" : "2016-06-27T15:11:00.000+03:00",
    "ArchiveTaskId" : "104b38d4-07d7-4d2f-84da-49b3e255d2bf",
    "ErrorMessage" : "null",
    "State" : "ReadyForDownload"
}
```

4.15.6.2.5 Параметры ответа:

Параметр	Описание
Percents	Процент выполнения задачи
Url	Ссылка на mp4-файл. Требуемый файл mp4 можно скачать, используя эту URL ссылку
CameraId	Идентификатор камеры
From	Время начала фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС
To	Время окончания фрагмента архива в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС
ArchiveTaskId	Идентификатор задачи
ErrorMessage	Сообщения об ошибках
State	Результат выполнения задачи

4.15.6.3 Удаление архива

4.15.6.3.1 Общий формат запроса:

DELETE http://IP-адрес:порт/removearchive?archivetaskid={archivetaskid}

4.15.6.3.2 Параметры запроса:

Параметр	Обязательный	Описание
archivetaskid	Да	Идентификатор задачи

4.15.6.3.3 Пример запроса:

DELETE http://127.0.0.1:80/removearchive?archivetaskid=084b56a5-bd49-4327-82db-9bc911f7ff96

4.15.6.3.4 Пример ответа:

```
{
    "ArchiveTaskId" : "084b56a5-bd49-4327-82db-9bc911f7ff96",
    "ErrorMessage" : null,
    "Success" : true
}
```

4.15.6.3.5 Параметры ответа:

Параметр	Описание
ArchiveTaskId	Идентификатор задачи
ErrorMessage	Сообщения об ошибках
Success	Результат удаления архива

Из папки экспорта удалится соответствующая папка с mp4-файлом.

4.15.7 Управление функциями средства видеонаблюдения

> **Внимание!**
>
> Для изменения настроек устройства приведенными ниже командами необходимо, чтобы была отключена функция **Использовать настройки устройства** – см. Панель настройки объекта Устройство видеоввода.
>
> Для управления параметрами с помощью команд ЕЦХД необходимо, чтобы сама камера поддерживала соответствующие команды (как при подключении по ONVIF, так и при использовании конкретного драйвера).

4.15.7.1 Общий формат запроса:

GET https://IP-адрес:порт/?cameraID={cameraID}&ip={ip}&loqin={loqin}&pass={pass}&action={action}&x={x}&y={y}&z={z}
&modelName={modelName}

4.15.7.2 Параметры запроса

Параметр	Обязательный	Описание
1	Да	Идентификатор средства видеонаблюдения (камеры).
2	Нет	IP-адрес средства видеонаблюдения.

Параметр	Обязательный	Описание
3	Нет	Учетная запись средства видеонаблюдения.
4	Нет	Пароль доступа к средству видеонаблюдения.
5	Да	Имя команды: **degreesmove** – дискретное движение. Атомарный сдвиг средства видеонаблюдения в указанном направлении. **degreesmove2** – относительное движение. Поворот средства видеонаблюдения относительно текущего положения. Область видимости средства видеонаблюдения делится на сетку, где центральная точка имеет координаты (x:0, y:0), левая верхняя (x:-7, y:7), правая нижняя (x:7, y:-7). Поворот средства видеонаблюдения должен быть осуществлен таким образом, чтобы объект по указанным в команде координатам оказался в центре изображения средства видеонаблюдения. Допускается «оптическая» погрешность, возникающая в результате расстояния до объекта видимости. Погрешность, возникающую за счет проекции сферы на плоскость, следует компенсировать. В зависимости от камеры для корректной работы данной команды может быть необходимо установить ключи реестра: 1. Камера не поддерживает Point&Click, но поддерживает абсолютные координаты. Ключу реестра ReplacePointAndClick должно быть установлено значение 1 (см. Справочник ключей реестра). 2. Камера поддерживает Point&Click. Ключу реестра ReplacePointAndClick должно быть установлено значение 0, а ключу TelemetryCommandMoveTimeout значение задержки между поворотом и масштабированием в миллисекундах (см. Справочник ключей реестра). **setposition** – установка положения средства видеонаблюдения. Перевод средства видеонаблюдения в указанное положение в градусах относительно «0» позиции. **getposition** – получение положение средства видеонаблюдения в плоскостях PAN и TILT в градусах, а также текущие значение зума. **focus** – команда фокусировки средства видеонаблюдения, где параметр z управляет поведением фокуса: • 1: Увеличить фокус • -1 : Уменьшить фокус • 0: Авто **iris** – команда управления диафрагмой средства видеонаблюдения, где параметр z управляет поведением диафрагмы: • 1: Открыть диафрагму • -1 : Закрыть диафрагму • 0: Авто **switch_day_night** – переключение ночного режима, где параметр z управляет режимом работы средства видеонаблюдения: • 1: Дневной режим • -1 : Ночной режим **backlight** – переключение подсветки, где параметр z управляет режимом работы подсветки: • 1: Включить • -1: Выключить **switch_color** – переключение черно-белого режима, где параметр z управляет режимом работы средства видеонаблюдения: • 1: Включить • -1: Выключить

Параметр	Обязательный	Описание
6	Нет	В командах **degreesmove, setposition** – поворот в плоскости PAN [-180 ..0.. 180]. В команде **degreesmove2** – поворот в плоскости PAN [-7..0..7]. В команде **getposition** не используется. В командах **focus, iris, switch_day_night, backlight, switch_color** следует задавать параметру значение 0.
7	Нет	В команде **degreesmove, setposition** – поворот в плоскости TILT [-180 ..0.. 180]. В команде **degreesmove2** – поворот в плоскости TILT [-7..0..7]. В команде **getposition** не используется. В командах **focus, iris, switch_day_night, backlight, switch_color** следует задавать параметру значение 0.
8	Нет	В командах **degreesmove, setposition** – увеличение/уменьшение зума [0.. 100]. В команде **degreesmove2** – увеличение/уменьшение зума [-1..0..1]. В команде **getposition** не используется. В командах **focus, iris, switch_day_night, backlight, switch_color** – задает режим работы средства видеонаблюедния, см. описание соответствующей команды.
9	Нет	Модель средства видеонаблюдения.

4.15.7.3 Пример запроса:

GET http://127.0.0.1:80/execute?cameraID=7&action=getposition

4.15.7.4 Пример ответа:

Ответ приходит только на команду getposition. Пример в формате JSON:

```
{"y":56, "x":105, "z":0}
```

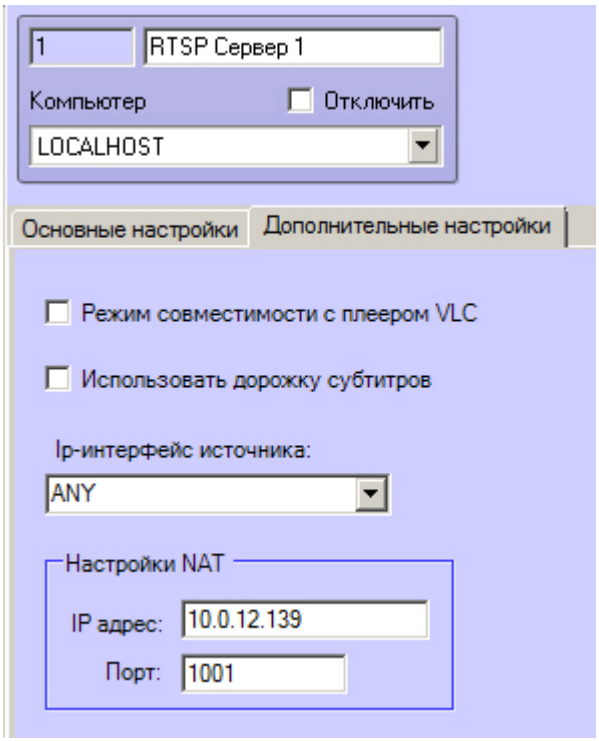
4.15.7.5 Параметры ответа:

Параметр	Описание
x	Координата в плоскости PAN
y	Координата в плоскости TILT
z	Значение масштабирования (зума)

4.15.8 Пример использования команд ЕЦХД при работе с NAT

Включение и конфигурирование использования NAT осуществляется на панели настройки объектов RTSP Сервер и Web сервер (см. Настройка модуля RTSP Сервер и Включение обработки запросов ЕЦХД и выбор RTSP сервера).

Например, на панели настройки объекта **RTSP Сервер** указаны следующие параметры NAT:



10.0.12.139 – IP-адрес маршрутизатора.

1001 – порт маршрутизатора.

127.0.0.1 – адрес Сервера в его локальной подсети.

559 – порт RTSP Сервера.

80 – порт Web Сервера.

4.15.8.1 Общий формат запроса:

GET http://IP-адрес:порт/getliveurl?cameraid={cameraid}

4.15.8.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры

4.15.8.3 Пример запроса:

```
GET http://127.0.0.1:80/getliveurl?cameraid=1
```

4.15.8.4 Пример ответа:

Если флажок **Использовать NAT-адрес** на панели настройки объекта **Web сервер** снят:

```
{
        "rtspurl": "rtsp://127.0.0.1:559/archive?id=1"
}
```

Если флажок **Использовать NAT-адрес** на панели настройки объекта **Web сервер** установлен:

```
{
        "rtspurl": "rtsp://10.0.12.139:1001/archive?id=1"
}
```

4.15.8.5 Параметры ответа:

Параметр	Описание
id	Идентификатор камеры

4.16 Отправка реакций, событий и xml-данных в ПК Интеллект через HTTP API

4.16.1 Отправка реакций в ПК Интеллект через HttpListener

Для того, чтобы использовать функцию отправки реакций в ПК *Интеллект* через HttpListener, на вкладке **Оборудование** диалогового окна **Настройка системы** на базе объекта **Компьютер** должен быть создан объект **Устройство видеоввода**, в поле **Тип** которого выбрано **HttpListener**, а в поле **Порт** указан порт подключения. Также на базе данного объекта **Устройство видеоввода** должно быть создано не более 4 объектов **Луч**.

HttpListener позволяет отправить только реакцию на закрытие/открытие нормально открытого/закрытого луча. После выполнения команды через 2 секунды луч будет переведен в нормальное состояние. В ПК *Интеллект* можно настроить макрокоманды или скрипты с использованием события о сработке луча.

4.16.1.1 Общий формат запроса:

POST http:/IP-адрес:порт/device/id/{id}

```
{"state": "{state}"}
```

4.16.1.2 Параметры запроса:

Параметр	Обязательный	Описание
Порт	Да	Порт HttpListener
id	Да	Идентификатор луча: 0/1/2/3
state	Да	Состояние луча: • **opened** - Открыть • **closed** - Закрыть

4.16.1.3 Пример запроса:

POST http://127.0.0.1:8085/device/id/0

```
{"state": "closed"}
```

4.16.2 Отправка реакций и событий в ПК Интеллект по HTTP-запросу

При получении команд описанного вида в ПК *Интеллект* будут генерироваться обычные события и реакции, которые можно по необходимости использовать в скриптах и макрокомандах (см. Руководство Администратора, раздел Создание и использование макрокоманд, а также Руководство по программированию (JScript)).

4.16.2.1 Общий формат запроса:

> **Примечание.**
>
> Для работы запросов необходимо создать объект Web-сервер – см. Создание объекта Web сервер

GET http://IP-адрес:порт/intellect_core/React?command="{react}"
GET http://IP-адрес:порт/intellect_core/Event?command="{event}"

или (аналогично)

POST http://IP-адрес:порт/intellect_core/React HTTP/1.1

```
{
        "command" : "{react}"
}
```

POST http://IP-адрес:порт/intellect_core/Event HTTP/1.1

```
{
        "command" : "{event}"
}
```

4.16.2.2 Параметры запроса:

Параметры	Обязательный	Описание
command	Да	React – команда в формате ПК *Интеллект* Event – событие в формате ПК *Интеллект*

4.16.2.3 Примеры запроса:

Добавление субтитров на видеоизображение с камеры 2 при помощи HTTP-запроса:

```
GET http://127.0.0.1:8080/intellect_core/React?command="CAM|2|ADD_SUBTITLES|command<Some
text\n!>"
```

Сгенерировать тревогу по камере 2 при помощи HTTP-запроса:

```
GET http://127.0.0.1:8080/intellect_core/Event?command="CAM|2|MD_START"
```

ИЛИ (аналогично)

```
POST http://127.0.0.1:8080/intellect_core/Event HTTP/1.1
{
"command" : "CAM|2|MD_START"
}
```

Вызов макрокоманды 1 при помощи HTTP-запроса:

```
GET http://127.0.0.1:8080/intellect_core/React?command="MACRO|1|RUN"
```

Начать запись по Камере 1:

```
POST http://127.0.0.1:8080/intellect_core/React HTTP/1.1
{
"command" : "CAM|1|REC"
}
```

4.16.3 Пример отправки команд HTTP API из утилиты curl

Для тестирования работы HTTP API можно отправлять команды HTTP API из утилиты curl. Данная свободно распространяемая утилита доступна для загрузки на официальном сайте https://curl.haxx.se/.

Для использования утилиты необходимо запустить командную строку Windows и перейти в папку <Директория установки curl>\curl-7.46.0-win64\bin.

4.16.3.1 Пример запроса:

Команда создания задания на экспорт архива (см. Экспорт архива):

curl -H "Content-Type: application/json" -X POST -d "{ \"CameraId\": \"1\", \"From\": \"2017-12-26T10:58:00.00Z\", \"To\": \"2017-12-26T11:00:00.00Z\" } " http://127.0.0.1:80/createarchivetask

4.16.3.2 Пример ответа:

```
{
        "CameraId" : "1", "From" : "2017-12-26T10:58:00.00Z", "To" : "2017-12-26T11:00:00.00Z",
"ArchiveTaskId" : "084b56a5-bd49-4327-82db-9bc911f7ff96", "ErrorMessage" : null, "State" :
"Created"
}
```

4.16.4 Отправка xml-файла

HTTP API позволяет получать данные в формате xml для дальнейшей обработки в скриптах. Для отправки файла необходимо выполнить POST запрос.

4.16.4.1 Общий формат запроса:

POST https://IP-адрес:порт/intellect_core/{Any}

```
<tag1>
    <tag2>some_data</tag2>
    <tag3>another_data</tag3>
</tag1>
```

4.16.4.2 Параметры запроса:

Параметр	Обязательный	Описание
Any	Да	Может быть указан любой набор допустимых символов, кроме Event и React, например: • https://127.0.0.1:8080/intellect_core • https://127.0.0.1:8080/intellect_core/Any • https://127.0.0.1:8080/intellect_core/Custom

4.16.4.3 Пример запроса:

POST https://127.0.0.1:8080/intellect_core/Any

```
<tag1>
    <tag2>some_data</tag2>
    <tag3>another_data</tag3>
</tag1>
```

4.16.4.4 Пример ответа:

После получения данных в формате xml в ПК *Интеллект* будет сгенерировано событие вида:

```
HTTP|1|CUSTOM_EVENT|url</intellect_core/Any>,owner<SLAVE-ID>,data<PG5..90ZT4=>
```

4.16.4.5 Параметры ответа:

Параметр	Описание
data	Тело запроса (xml в примере выше) в кодировке base64
url	Часть посланного url

4.16.5 Получение кадра архива

4.16.5.1 Общий формат запроса:

http://IP-адрес:порт/video/GetImage/{cameraid}/{UTC_Time}

4.16.5.2 Параметры запроса:

Параметр	Обязательный	Описание
cameraid	Да	Идентификатор камеры
UTC_Time	Да	Время в формате UTC

4.16.5.3 Пример запроса:

Получение кадра архива (скриншота) с камеры 1, дата 07.11.2018, время 13:20:56.212:

`http://127.0.0.1:80/video/GetImage/1/20181107T132056.212`

4.16.5.4 Пример ответа:

В ответ придет либо изображение в формате jpeg, либо HTTP ERROR 404

4.17 Настройка интеграции с Техносерв

Для работы интеграции с Техносерв необходимо создать и настроить следующие объекты и файлы:

1. **Web-сервер** – см. Настройка Сервера для подключения Клиентов с помощью модуля Web-сервер
2. **Веб-сервер 2.0** – см. Настройка Сервера для подключения Клиентов с помощью модуля Веб-сервер 2.0
3. Хотя бы один объект **Пользователь**, добавленный к **Правам пользователя**. Под учетными данными данного пользователя будет происходить подключение к ПК Интеллект. См. Администрирование прав и полномочий.
4. На панели настройки объекта **Веб-сервер 2.0** необходимо включить передачу событий от камеры и от детекторов – см. Настройка фильтра событий для модуля Веб-сервер 2.0
5. Все камеры, по которым требуется получать события, добавить в настройки объекта **Web-сервер** – см. Выбор и настройка видеокамер для Web-сервера
6. Заполнить конфигурационный файл ApiBgConfig.xml и поместить его в папку "<Директория установки ПК *Интеллект*>\Modules\Jetty\" – см. Заполнение конфигурационного файла ApiBgConfig.xml

См. также Примеры команд для работы с интеграцией Техносерв.

4.17.1 Заполнение конфигурационного файла ApiBgConfig.xml

Файл ApiBgConfig.xml предназначен для настройки логина и пароля для подключения системы Техносерв к ПК *Интеллект*, а также для сопоставления событий ПК *Интеллект* кодам событий Техносерв (см. таблицу ниже).

Файл имеет следующий формат:

```
<ApiBgEventsConfiguration>
    <PingTimeout>120000</PingTimeout>
    <MainHost>127.0.0.1</MainHost>
    <Login></Login>
    <Password></Password>
    <ApiBgEventMatch>
        <ProductObjectName>CAM</ProductObjectName>
        <ProductEventName>DETACHED</ProductEventName>
        <ApiBgEventCode>not_available</ApiBgEventCode>
    </ApiBgEventMatch>
    <ApiBgEventMatch>
        <ProductObjectName>CAM</ProductObjectName>
        <ProductEventName>ATTACH</ProductEventName>
        <ApiBgEventCode>available</ApiBgEventCode>
        <Picture>1</Picture>
    </ApiBgEventMatch>
    <ApiBgEventMatch>
        <ProductObjectName>CAM</ProductObjectName>
        <ProductEventName>ALARM</ProductEventName>
        <ApiBgEventCode>264400</ApiBgEventCode>
        <Shapes>1</Shapes>
        <Picture>1</Picture>
    </ApiBgEventMatch>
        <ApiBgEventMatch>
        <ProductObjectName>CAM</ProductObjectName>
        <ProductEventName>REC</ProductEventName>
        <ApiBgEventCode>264409</ApiBgEventCode>
        <Shapes>1</Shapes>
        <Picture>1</Picture>
    </ApiBgEventMatch>
</ApiBgEventsConfiguration>
```

1. Время в миллисекундах, между отправкой PING-событий
   <PingTimeout>120000</PingTimeout>
2. IP-адрес, который необходимо возвращать в ответ на запросы системы Техносерв.
   <MainHost>127.0.0.1</MainHost>
3. Логин и пароль для подключения к серверу Техносерв и отправки событий (если есть)
   <Login></Login> <Password></Password>
4. Описание соответствия события ПК *Интеллект* и кода события:

```
<ApiBgEventMatch>
    <ProductObjectName>CAM_VMDA_DETECTOR</ProductObjectName>
    <ProductEventName>ALARM</ProductEventName>
    <ApiBgEventCode>264400</ApiBgEventCode>
    <ProductObjectIds>
      <id>2</id>
      <id>1</id>
    </ProductObjectIds>
    <ParamMatch>
        <ProductEventParamName>num</ProductEventParamName>
       <ApiBgEventParamCode>NUMBER_OF_DETECTED_PEOPLE</ApiBgEventParamCode>
    </ParamMatch>
    <Shapes>1</Shapes>
    <Picture>1</Picture>
</ApiBgEventMatch>
```

   a. ProductObjectName, ProductEventName – объект и событие ПК *Интеллект*, например, CAM|MD_START.
   b. ProductObjectIds – id объектов Интеллекта, если требуется задать конкретные детекторы. Может отсутствовать, тогда будут передаваться события по всем детекторам.
   c. ApiBgEventCode – код события, см. ниже.
   d. Shapes – нужно ли передавать поле Shapes.
   e. Picture – нужно ли передавать ссылку на стоп-кадр.
   f. ParamMatch - соответствие имени параметра события Интеллекта параметру события. Может отсутствовать.

Коды событий:

Код	**Наименование события**
1799	Детектор толпы (Обнаружение скопления людей, в том числе в несанкционированных местах)
264393	Детектор оставленных предметов
264383	Подсчет количества людей
264385	Оценка плотности потока людей на значимых для города объектах
264388	Выявление движения человека против потока
264387	Остановка человека в контролируемой зоне (с заданием времени остановки)
264386	Резкое ускорение движения человека
264389	Хаотичное движение человека (праздношатание)
264391	Индексирование событий в условиях дорожного движения (заторы)

Код	Наименование события
264392	Индексирование событий в условиях дорожного движения (массовое скопление автотранспорта)
264390	Индексирование событий в условиях дорожного движения (плотность потока)
264394	Детектор фактов пересечения запрещенной зоны (проезд, проход)
264395	Детектор исчезнувших предметов
264396	Реагирование на проход людей в заданном направлении (вход)
264397	Реагирование на проход людей в заданном направлении (выход)
264399	Реагирование на проход людей в заданном направлении (коридор и т.п.)
264398	Реагирование на проход людей в заданном направлении (переходы)
264400	Появление человека или автомобиля в зоне наблюдения (улицы, площади, перекрестки, парки)
1795	Детектор расфокусировки
1797	Загрязнение объектива видеокамеры
264403	Изменение фона
264402	Сдвиг камеры
1796	Детектор засветки/заслонения
264401	Открытое пламя
287069	Обнаружена цель
siren	Зафиксирована сирена или автомобильная сигнализация
shout	Зафиксирован крик человека
klaxon	Зафиксирован звук клаксона автомобиля
voice	Зафиксирована речь человека
brake	Зафиксирован звук автомобильного тормоза
shockwave	Зафиксирована ударная волна
shock	Зафиксирован хлопок/выстрел/взрыв

Код	Наименование события
PING	Используется для регулярного события с заданной на стороне ИС периодичностью для контроля состояния канала связи между ИС и АПК БГ. Значение заполняется cameraCode=-1.
not_available	Камера недоступна (отсутствует видеопоток или нет связи с камерой и т.п.)
available	Камера доступна

4.17.2 Примеры команд для работы с интеграцией Техносерв

4.17.2.1 Запрос на получение списка камер

4.17.2.1.1 Общий формат запроса**:**

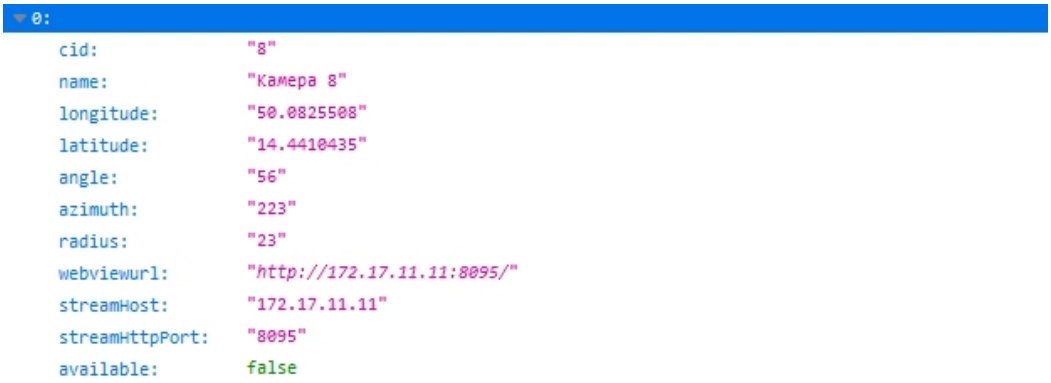
GET http://IP-адрес:порт/web2/secure/api/bg

4.17.2.1.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/api/bg

> **Примечание.**
>
> Если в поле **Дополнительная информация** на панели настройки камеры указать координаты камеры в формате coords:[longitude]:[latitude]:[angle]:[azimuth]:[radius], то эти данные будут возвращены в ответе на запрос в соответствующих параметрах. См. также Настройка дополнительной информации о камере.

4.17.2.1.3 Пример ответа:



4.17.2.2 Запрос на получение списка подписок

4.17.2.2.1 Общий формат**:**

GET http://IP-адрес:порт/web2/secure/api/bg/events/subscriptions

4.17.2.2.2 Пример запроса:

GET http://127.0.0.1:8085/web2/secure/api/bg/events/subscriptions

4.17.2.2.3 Пример ответа:

Если список пустой:

{"data": [], "status":"success"}


```
data:     []
status:   "success"
```


при наличии подписок:


```
data:
  0:
    callback:    "https://webhook.site/4589bc86-e597-41d3-aab8-a93b09e977a8"
    filter:
      action:    "RUN"
      type:      "MACRO"
      id:        "8"
    id:          "63eb2852-9780-4085-bf5e-f3e5be875357"
status:          "success"
```


4.17.2.3 Запрос на создание подписки

4.17.2.3.1 Общий формат:

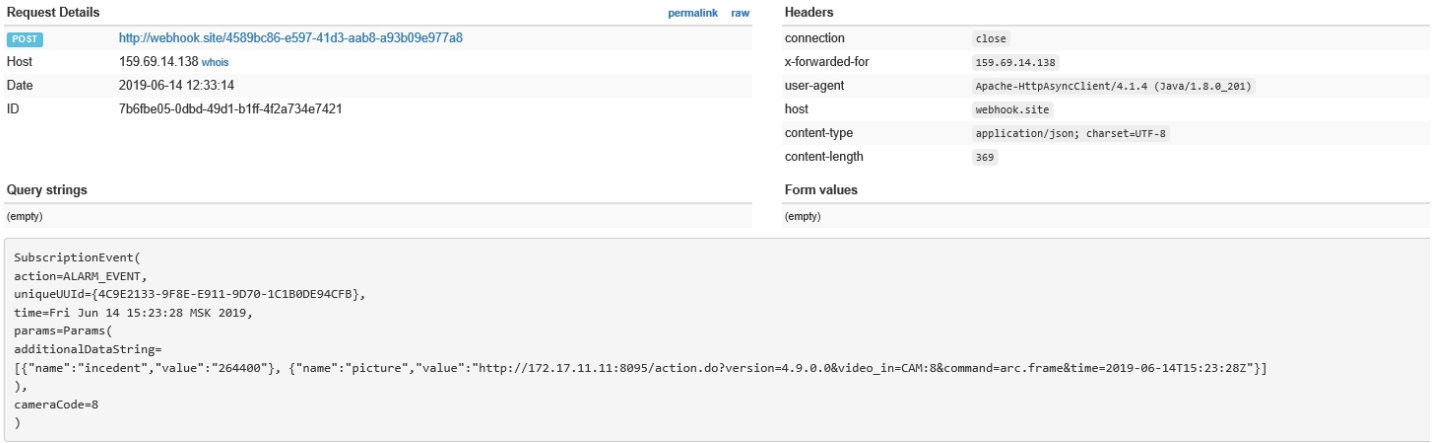
http://IP-адрес:порт/web2/secure/api/bg/events/subscriptions

4.17.2.3.2 Пример запроса:

POST http://127.0.0.1:8085/web2/secure/api/bg/events/subscriptions

{"callback":"https://webhook.site/26d15078-c405-4918-8f03-4f2a01b7580f","filter":{"action":"RUN","type":"MACRO"},"id":"5"}

4.17.2.3.3 Пример ответа:



4.17.2.4 Запрос на удаление одной подписки

4.17.2.4.1 Общий формат:

DELETE http://IP-адрес:порт/web2/secure/api/bg/events/subscriptions/{sub_id}

4.17.2.4.2 Параметры запроса:

Параметр	Обязательный	Описание
sub_id	Да	Идентификатор подписки

4.17.2.4.3 Пример запроса:

DELETE http://127.0.0.1:8085/web2/secure/api/bg/events/subscriptions/071e0159-5b2c-4bab-8ed7-42397f1b99b8

4.17.2.5 Запрос на удаление всех подписок

4.17.2.5.1 Общий формат**:**

DELETE http://IP-адрес:порт/web2/secure/api/bg/events/subscriptions/all

4.17.2.5.2 Пример запроса:

DELETE http://127.0.0.1:8085/web2/secure/api/bg/events/subscriptions/all

5 HTTP-сервер ПК Интеллект

5.1 Общие сведения о HTTP Сервере

HTTP Сервер позволяет передавать события из ПК *Интеллект* в сторонние системы, т.е. оповещать клиенты о событиях.

Для работы с модулем *HTTP Сервер* необходимо, чтобы веб-браузер поддерживал технологию Server-sent events (SSE).

> **Примечание.**
>
> Браузер Internet Explorer не поддерживает технологию SSE.

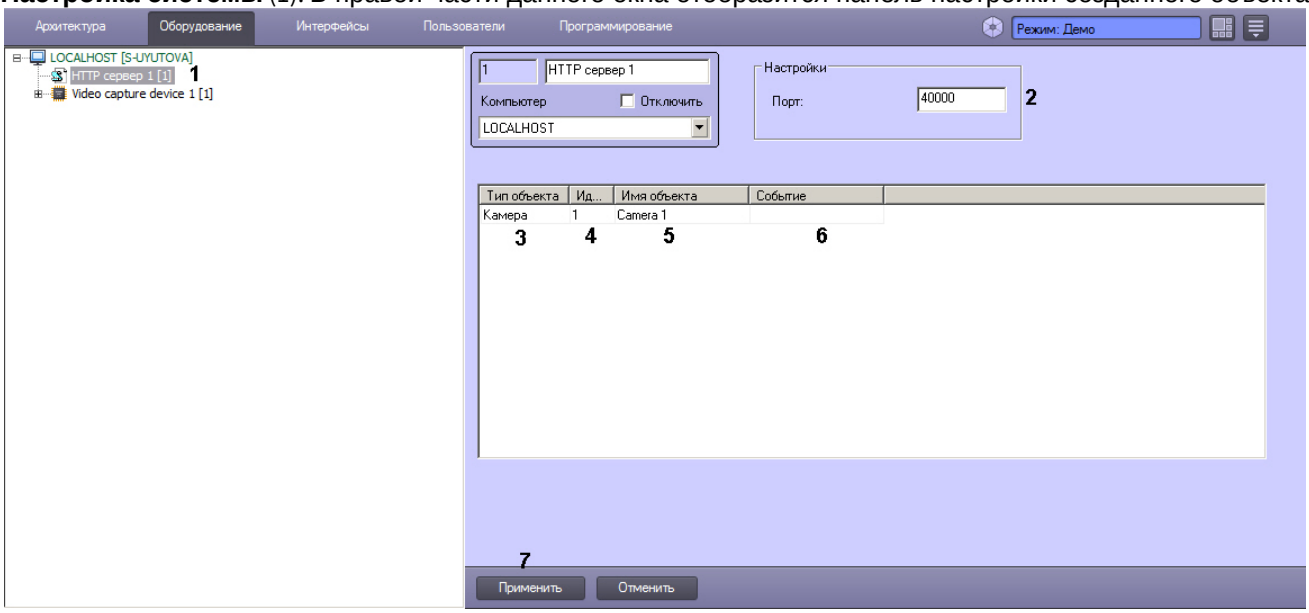
Настройка модуля *HTTP Сервер* осуществляется на панели настройки соответствующего объекта – см. Настройка объекта HTTP Сервер.

Для работы с модулем *HTTP Сервер* используются запросы из веб-браузера – см. Запросы к HTTP Серверу.

5.2 Настройка объекта HTTP Сервер

Настройка объекта **HTTP Сервер** осуществляется в следующем порядке:

1. Создать объект **HTTP Сервер** на базе объекта **Компьютер** на вкладке **Оборудование** диалогового окна **Настройка системы** (1). В правой части данного окна отобразится панель настройки созданного объекта.



2. В поле **Порт** указать номер порта в системе, на который клиент должен отправлять запросы к *HTTP Серверу* (2).
3. Настроить фильтр событий, информация о которых должна пересылаться *HTTP Сервером*. Если в фильтр не добавлены никакие события, то *HTTP Сервер* возвращает на запросы клиентов только события ядра (объекта CORE). Выбор дополнительных событий осуществляется следующим образом:
   a. Из раскрывающегося списка **Тип объекта** выбрать требуемый тип объекта ПК *Интеллект* (3).
   b. Из раскрывающегося списка **Идентификатор** выбрать идентификатор объекта выбранного типа (4). Если идентификатор не выбран, то *HTTP Сервер* будет отправлять события всех объектов выбранного типа.
   c. После выбора типа и идентификатора объекта поле **Имя объекта** будет автоматически заполнено названием соответствующего объекта (5).

d. Из раскрывающегося списка **Событие** выбрать тип события для отправки через HTTP Сервер (6). Если событие не выбрано, то *HTTP Сервер* будет отправлять все события выбранного объекта (объектов выбранного типа).
e. Повторить шаги a-d для всех требуемых объектов и событий.

4. Нажать на кнопку **Применить** (7).

Настройка объекта **HTTP Сервер** завершена.

5.3 Запросы к HTTP Серверу

Для работы с модулем *HTTP Сервер* используются запросы, описанные ниже.

Запрос идентификатора последнего события

```
http://<IP-адрес>:<Порт>/core/GetLastID
```

Пример запроса

```
http://localhost:40000/core/GetLastID
```

Пример ответа

```
{"lastId":"b686658c-764c-e911-8f42-001a7dda710e"}
```

Запрос событий с учетом фильтра

Фильтр настраивается на панели настройки объекта HTTP Сервер – см. Настройка объекта HTTP Сервер.

```
http://<IP-адрес>:<Порт>/core/EventsTest
```

Пример запроса

```
http://localhost:40000/core/EventsTest
```

Пример ответа

```
{"module":"video.run","protocol_id":"2dc6dfcb-5351-e911-8832-534e57000000","slave_id":"S-
UYUTOVA","src_action":"MD_STOP","src_objid":"2","src_objtype":"CAM","time":"2019-03-28T12
:20:03.977"}
```

6 Настройка RabbitMQ

ПК *Интеллект* может выступать в роли приемника и передатчика событий RabbitMQ. Для использования данной функциональности необходимо предварительно загрузить с веб-сайта https://www.rabbitmq.com/install-windows.html и установить следующие компоненты:

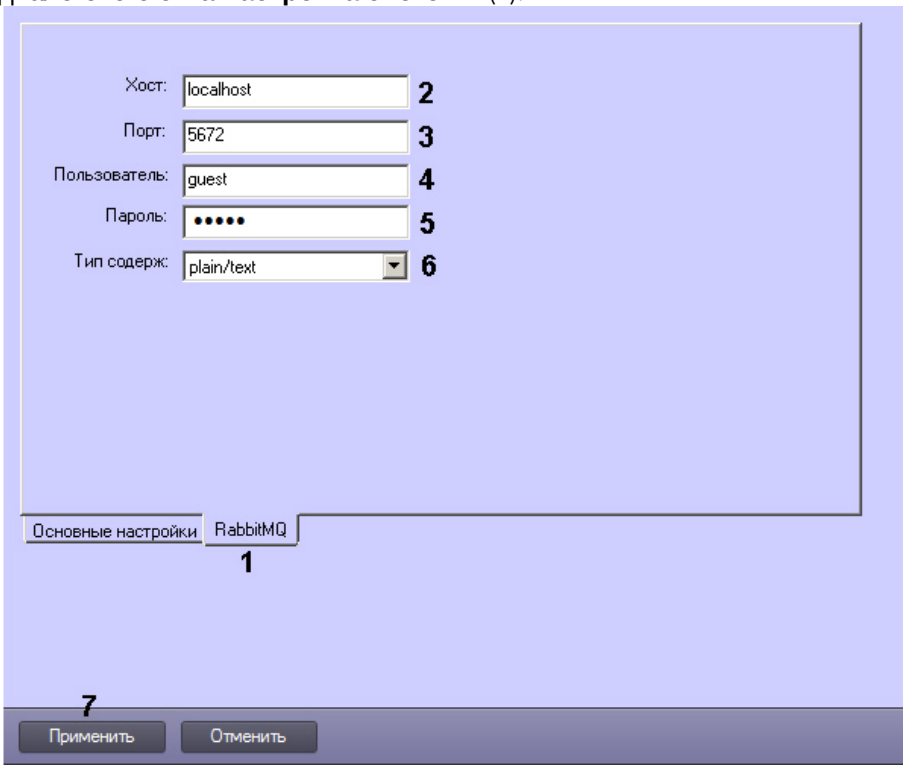
1. Rabbit-сервис
2. Erlang

По умолчанию RabbitMQ имеет встроенную учетную запись **guest/guest**, которая работает только с хостом **localhost**. Если требуется использовать RabbitMQ удаленно, необходимо завести новую учетную запись.

В примерах ниже упоминаются утилиты amqp_sendstring.exe и amqp_listen.exe. Данные утилиты для обработки и отправки сообщений ПК *Интеллект* программисту необходимо реализовать самостоятельно.

Настройка подключения к RabbitMQ осуществляется следующим образом:

1. Перейти на вкладку **RabbitMQ** на панели настройки **Объект охраны** на вкладке **Программирование** диалогового окна **Настройка системы** (1).



2. В поле **Хост** ввести имя или адрес хоста RabbitMQ (2).
3. В поле **Порт** ввести порт для подключения к хосту RabbitMQ (3). Если порт не задан (по умолчанию) или равен 0, то клиент RabbitMQ не запускается.
4. В поле **Пользователь** ввести имя пользователя хоста RabbitMQ (4).
5. В поле **Пароль** ввести пароль пользователя хоста RabbitMQ (5).
6. Из раскрывающегося списка **Тип содерж.** выбрать тип данных, отправляемых и принимаемых ПК *Интеллект* (6):
   a. **application/json** – отправляется и принимается сообщение в формате JSON. Если ПК *Интеллект* не может разобрать сообщение в формате JSON, то предпринимается попытка распознать сообщение в формате **plain/text**.
   b. **plain/text** – отправляется и принимается событие ПК Интеллект в формате "ТИП|ИДЕНТИФИКАТОР|СОБЫТИЕ"
7. Нажать на кнопку **Применить** (7).

6.1 Поставщик ПК Интеллект и сторонний приемник

Поставщик ПК *Интеллект* работает в режиме «amq.topic», а приемник – в режиме “amq.direct”.

amq.topic позволяет ранжировать подписки. Каждое событие, отправляемое из ПК *Интеллект*, подписывается специальным заголовком, который строится по следующей схеме:
routingkey = "intellect.event." + msg.GetSourceType() +"." + msg.GetSourceId() + "." + msg.GetAction();

Таким образом, приемник может гибко подписаться на необходимые события.

Пример.

Подписка на все события:

```
amqp_listen.exe localhost 5672 amq.topic intellect.event.#
```

Подписка только на события с Action==”RUN”

```
amqp_listen.exe localhost 5672 amq.topic intellect.event.*.*.RUN
```

> **Внимание!**
>
> События поступают в очередь RabbitMQ от каждого ядра индивидуально. Например, если в системе два ядра, и событие произошло на ядре, у которого неправильные настройки или нет подключения к RabbitMQ, событие не попадет в очередь, несмотря на то, что второе ядро его получит. Каждое ядро отправляет только свои сообщения.

6.2 Приемник ядра ПК Интеллект

Приемник реализован по схеме amq.direct. Это означает, что он подписан на все события с ключом "bindingkey". bindingkey реализован по схеме "intellect." + ComputerName (чувствителен к регистру).

При отправке bindingkey должен быть указан в точности как у приемника, иначе сообщение не дойдет.

Таким образом, чтобы ядро получило событие, необходимо отправить следующее сообщение:

текстовое

```
amqp_sendstring.exe localhost 5672 amq.direct intellect.ASUS "CAM|1|HELLO"
```

или JSON

```
amqp_sendstring.exe localhost 5672  amq.direct intellect.ASUS {\"Type\":\"MACRO\",\"Id\":
\"1\",\"Action\":\"RUN\",\"Params\":{\"test1\":\"+++\",\"test2\":\"000\"}}
```

7 Заключение

Более подробная информация о программном комплексе *Интеллект* содержится в следующих документах:

1. Руководство администратора;
2. Руководство оператора;
3. Руководство по установке и настройке компонентов охранной системы;
4. Руководство по программированию;
5. Руководство по программированию (JScript).

Если в процессе работы с данным программным продуктом у вас возникли трудности или проблемы, вы можете связаться с нами. Однако рекомендуем предварительно сформулировать ответы на следующие вопросы:

1. В чем именно заключается проблема?
2. Когда и после чего появилась данная проблема?
3. В каких именно условиях проявляется проблема?

Помните, что чем более полную и подробную информацию вы нам предоставите, тем быстрее наши специалисты смогут устранить вашу проблему.

Мы всегда работаем над улучшением качества своей продукции, поэтому будем рады любым вашим предложениям и замечаниям, касающимся работы нашего программного обеспечения, а также документации к нему.

Пожелания и замечания по данному Руководству следует направлять в Отдел технического документирования компании Ай-Ти-Ви групп (documentation@itv.ru).

8 ПРИЛОЖЕНИЕ 1. Описание структуры ddi-файла

Описание полей таблицы, расположенной на вкладке **Имена** (раздел **<Objects>**), приведено в таблице ниже.

Поле	Описание
Имя (<ObjectName>)	Идентификационное имя объекта
Название (<VisibleName>)	Отображаемое имя
Имя группы (<GroupName >)	Имя группы объектов. Используется для объединения объектов в группу в дереве настроек ПК *Интеллект*

Описание полей таблицы, расположенной на вкладке **События** (раздел **<Events>**), приведено в таблице ниже.

Поле	Описание
Название (<EventName>)	Идентификационное имя события.
Описание (<EventDescription>)	Описание сообщения, выводимое в протоколе событий.
Обработка сообщений (<EventType>)	Определение цвета фона сообщения в протоколе событий: обычное – без цвета; тревожное – красное окно; информационное – синее окно.
Звуковая поддержка (<IsSoundEnabled>)	Воспроизведение звукового файла при срабатывании сообщения.
Не слать в сеть (<IsNetworkDisabled>)	Не посылать сообщение по сети.
Не протоколировать (<IsProtocolDisabled>)	Не отображать сообщение в протоколе событий.
Журнал Windows (<IsWindowsLogEnabled>)	Сохранять сообщения в журнале событий Windows. *Примечание. Запись в журнал Windows невозможна, если событие не протоколируется*

Описание полей таблицы, расположенной на вкладке **Реакции** (раздел **<Reacts>**), приведено в таблице ниже.

Поле	Описание
Название (<ReactName>)	Имя реакции
Описание (<ReactDescription>)	Описание реакции, выводимое в контекстном меню при щелчке правой кнопкой мыши по значку объекта на *Карте*
Флаги (<IsReactArm>)	Флаг выполнения реакции: либо для одного объекта, либо для группы объектов, входящих в один раздел

Описание полей таблицы, расположенной на вкладке **Значки** (раздел **<Icons>**), приведено в таблице ниже.

Поле	Описание
Имя файла (<FileName>)	Часть имени bmp-файла, которая является идентификатором изображения. Идентификатор изображения позволяет использовать несколько bmp-файлов для представления на *Карте* объектов одного типа (см. раздел Использование утилиты ddi.exe для работы с DDI-файлами)

Название (<IconName>)	Описание bmp-файла объекта

Описание полей таблицы, расположенной на вкладке **Состояния** (раздел **<States>**), приведено в таблице ниже.

Поле	**Описание**
Название (<StateName>)	Имя состояния
Изображение (<ImgName>)	Часть имени, которая является идентификатором состояния, соответствующего bmp-файла (см. раздел Использование утилиты ddi.exe для работы с DDI-файлами). *Примечание. Объекты на Карте могут быть отображены с помощью линий, т.е. без использования bmp-файлов. В этом случае, если изменяется состояние объекта, меняется цвет линии. Цвет (RGB) состоянию задается следующим образом:* ***<Состояние>$R:G:B***
Описание (<StateDescription>)	Описание состояния
Мерцание (<IsStateFlashing>)	Отображение на *Карте*: обычное – отсутствие мерцания, тревожное – мерцание значка на *Карте*

Описание полей таблицы, расположенной на вкладке **Правила перехода** (раздел **<Rules>**), приведено в таблице ниже.

Поле	**Описание**
Событие (<EventName>)	Событие, по которому выполняется переход
Переход из состояния (<FromStateName>)	Исходное состояние, из которого должен быть осуществлен переход
Переход в состояние (<ToStateName>)	Итоговое состояние, в которое должен быть осуществлен переход

9 ПРИЛОЖЕНИЕ 2. Объявление классов NissObjectDLLExt и CoreInterface

На странице:

9.1 CoreInterface

```
class CoreInterface

{

public:

                virtual BOOL DoReact    (React&) = 0;

                virtual BOOL NotifyEvent(Event&) = 0;

                virtual void SetupACDevice(LPCTSTR objtype, LPCTSTR objid, LPCTSTR
 objtype_reader) = 0;



                virtual   BOOL    IsObjectExist(LPCTSTR objtype, LPCTSTR id) = 0;

                virtual   BOOL    IsObjectDisabled(LPCTSTR objtype, LPCTSTR id) = 0;

                virtual Msg FindPersonInfoByCard(LPCTSTR facility_code, LPCTSTR card) = 0;

                virtual Msg FindPersonInfoByExtID(LPCTSTR external_id) = 0;

                virtual   CString GetObjectName (LPCTSTR objtype, LPCTSTR id) = 0;

                virtual   CString GetObjectState(LPCTSTR objtype, LPCTSTR id) = 0;

                virtual   void       SetObjectState(LPCTSTR objtype, LPCTSTR id, LPCTSTR state)
= 0;

                virtual   BOOL    IsObjectState(LPCTSTR objtype, LPCTSTR id, CString state) =
0;



                virtual   CString GetObjectParam    (LPCTSTR objtype, LPCTSTR id, LPCTSTR
 param) = 0;

                virtual   int  GetObjectParamInt (LPCTSTR objtype, LPCTSTR id, LPCTSTR param) =
0;

                virtual   CMapStringToStringArray* GetObjectParamList(LPCTSTR objtype, LPCTSTR
 id, LPCTSTR param) = 0;

                virtual   CStringArray*   GetObjectParamList(LPCTSTR objtype, LPCTSTR id,
LPCTSTR param, LPCTSTR name) = 0;

                virtual   void       GetObjectParams   (LPCTSTR objtype, LPCTSTR id, Msg& msg)
= 0;

                virtual   void       SetObjectParamInt (LPCTSTR objtype, LPCTSTR id, LPCTSTR
 param, int val) = 0;
```

```
                virtual   CString GetObjectIdByParam(LPCTSTR type, LPCTSTR param, LPCTSTR val)
= 0;

                virtual   CString GetObjectIdByName(LPCTSTR type, LPCTSTR name) = 0;

                virtual   CString GetObjectParentId(LPCTSTR objtype, LPCTSTR id, LPCTSTR
 parent) = 0;

                virtual   int GetObjectIds(LPCTSTR objtype, CStringArray& list, LPCTSTR main_id
= NULL) = 0;

                virtual   int GetObjectChildIds(LPCTSTR objtype, LPCTSTR objid, LPCTSTR
 childtype, CStringArray& list) = 0;

};
```

9.2 NissObjectDLLExt

```
class NissObjectDLLExt

{

protected:

                CoreInterface* m_pCore;

public:

                NissObjectDLLExt(CoreInterface* core) { m_pCore = core; }


                virtual CString GetObjectType() = 0;

                virtual CString GetParentType() = 0;

                virtual int GetPos()  { return -1; }

                virtual CString GetPort() { return CString(); }

                virtual CString GetProcessName() { return CString(); }

                virtual CString GetDeviceType() { return CString(); }


                virtual BOOL HasChild() { return FALSE; }

                virtual UINT HasSetupPanel()  { return FALSE; }

                virtual void OnPanelInit(CWnd*) {}

                virtual void OnPanelLoad(CWnd*,Msg&) {}

                virtual void OnPanelSave(CWnd*,Msg&) {}

                virtual void OnPanelExit(CWnd*) {}

                virtual void OnPanelButtonPressed(CWnd*,UINT) {}

                virtual BOOL IsRegionObject()   { return FALSE; }

                virtual BOOL IsProcessObject()  { return FALSE; }

                virtual BOOL IsIncludeParentId()             { return 0; }

                virtual BOOL IsWantAllEvents()                                  { return 0; }

                virtual CString DescribeSubscribeObjectsList() { return CString(); }

                virtual CString GetIncludeIdParentType(){ return CString(); }
```

```
                virtual CString DescribeParamLists(){ return CString(); }


                virtual void OnCreate(Msg&) {}
                virtual void OnChange(Msg&,Msg&) {}
                virtual void OnDelete(Msg&) {}
                virtual void OnInit(Msg&)           {}
                virtual void OnEnable(Msg&)  {}
                virtual void OnDisable(Msg&) {}
                virtual BOOL OnEvent(Event&) { return FALSE; }
                virtual BOOL OnReact(React&) { return FALSE; }
};
```