Руководство по программированию

Обновлено 05.10.2023

Содержание

1 Руководство по программированию. Введение

2 Объект Программа. Программирование с использованием встроенного языка ПК Интеллект

3 Описание событий и реакций объектов системы

- GRABBER Устройство видеоввода
- CAM Камера
- MONITOR Монитор видеонаблюдения
- PLAYER Аудиопроигрыватель
- OLXA_LINE Микрофон
- DIALOG Окно запроса оператора
- MMS Сервис почтовых сообщений
- MAIL_MESSAGE Почтовое сообщение
- VMS Сервис голосовых сообщений
- GRELE Реле
- GRAY Луч
- VNS Сервис голосового оповещения
- SMS Сервис коротких сообщений
- TELEMETRY Поворотное устройство
- TELEMETRY_EXT Пульт управления
- MACRO Макрокоманда
- TIME_ZONE Временная зона
- SSS_WATCHDOG Служба перезагрузки системы
- SLAVE Компьютер
- DISPLAY Экран
- GATE Видеошлюз
- CAM_VMDA_DETECTOR Детектор VMDA
- ARCH Долговременный архив
- CORE Ядро
- TITLEVIEWER Поиск по титрам
- MAP Карта
- FAILOVER Сервис отказоустойчивости
- OPERATORPROTOCOL Протокол оператора
- PERSON Пользователь
- IPJOYSTICK Устройство управления
- CAM_FACECAPTURE Детектор лиц
- EVENT_VIEWER Протокол событий
- CAM_TITLE Титрователь
- IPSTORAGE Внешнее хранилище
- TELEGRAM Telegram бот
- BACNET BacNet
- CAM_IP_DETECTOR Детектор встроенный
- SIP_TERMINAL SIP-терминал
- INC_MANAGER Менеджер инцидентов
- INC_SERVER Сервер инцидентов

4 Заключение

5 Руководство по программированию. Введение

5.1 Назначение программного комплекса Интеллект

Программный комплекс *Интеллект* предназначен для создания промышленных масштабируемых гибко настраиваемых (адаптируемых) интегрированных систем безопасности на основе цифровых систем видеонаблюдения и аудиоконтроля.

Программный комплекс *Интеллект* обладает следующими основополагающими функциональными возможностями:

1. Интеграция цифровых систем видеонаблюдения и аудиоконтроля со смежными информационными системами, различного типа охранным оборудованием, вспомогательным программным обеспечением сторонних производителей при использовании интегрированных открытых интерфейсов информационного взаимодействия.
2. Совместимость с широким перечнем охранного оборудования и информационных систем безопасности, в частности, таких, как охранно-пожарная сигнализация, системы контроля доступа, видеокамеры, информационные системы анализа, распознавания и идентификации объектов (событий) на видеоизображении.
3. Централизованная регистрация и обработка событий, генерация оповещений и управляющих воздействий в соответствии с гибко настраиваемыми алгоритмами.
4. Практически неограниченные возможности масштабирования, адаптации к специфике решаемых задач, перераспределения используемых ресурсов при изменении количества или качества задач по мониторингу состояния подконтрольных объектов и управления различного рода оборудованием.

5.2 Настройка логических взаимосвязей между объектами в программном комплексе Интеллект

Функциональные возможности программного комплекса *Интеллект* основаны на логическом взаимодействии между объектами. Общие сведения о способах настройки логических взаимосвязей приведены в таблице.

Способ настройки логической взаимосвязи	**Описание**	**Реализация**	**Пример**
Панели настройки объектов системы	Базовая настройка взаимодействия между объектами системы	Реализуется с использованием функциональных возможностей объектов системы – см. Конфигурирование и настройка программного комплекса Интеллект	Настройка отображения видеосигнала в интерфейсном окне **Монитор**
Макрокоманда	Настройка простых взаимосвязей между объектами, функциональные возможности которых не позволяют выполнить требуемые операции	Реализуется с использованием функциональных возможностей объекта **Макрокоманда** – см. Создание и использование макрокоманд	Настройка включения исполнительного устройства (реле) при замыкании луча
Программа	Настройка комплексных взаимосвязей между объектами, если функциональные возможности объекта **Макрокоманда** не позволяют выполнить требуемые операции	Реализуется на базе объекта **Программа** в виде кода на встроенном в ПК *Интеллект* языке программирования – см. настоящее Руководство	Требуется каждые 15 минут возвращать поворотные камеры в исходное положение и делать снимок
Скрипт		Реализуется на базе объекта **Скрипт** в виде кода на языке JScript – см. документ Руководство по программированию (JScript)	

5.3 Назначение и структура руководства

Документ Руководство по программированию является справочно-информационным пособием по программированию на встроенном языке ПК *Интеллект* и предназначен для системных администраторов, специалистов по установке и настройке, пользователей с правами администрирования цифровых систем видеонаблюдения и аудиоконтроля, созданных на основе программного комплекса *Интеллект*.

Программирование в ПК *Интеллект* позволяет автоматизировать управление системой путем настройки комплексных логических взаимосвязей между объектами.

В данном Руководстве представлены следующие материалы:

1. инструментарий программирования;
2. описание синтаксиса встроенного языка программирования;
3. примеры программ на встроенном языке.

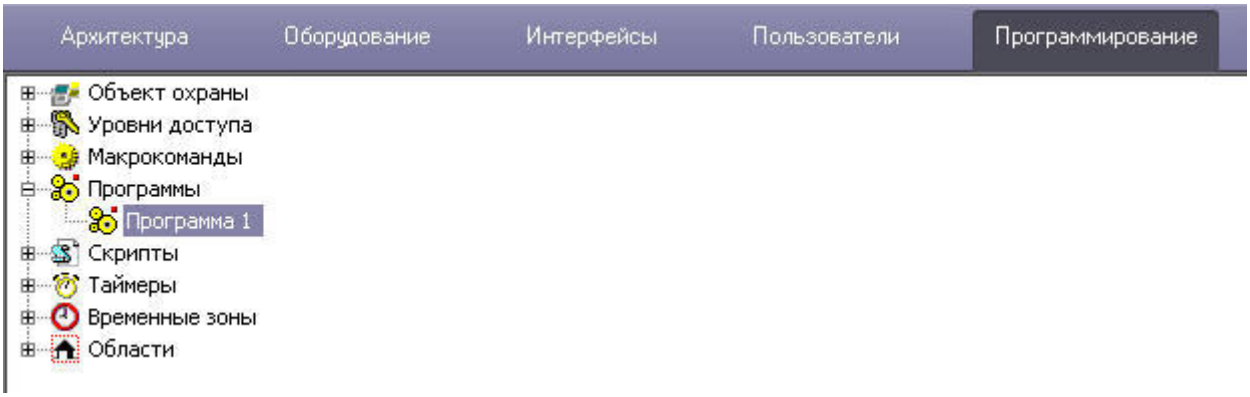
6 Объект Программа. Программирование с использованием встроенного языка ПК Интеллект

6.1 Инструментарий программирования

6.1.1 Системный объект Программа

Системный объект **Программа** предназначен для инициализации в ПК *Интеллект* программы, разработанной на собственном языке программирования ПК *Интеллект*, и задания параметров ее выполнения.

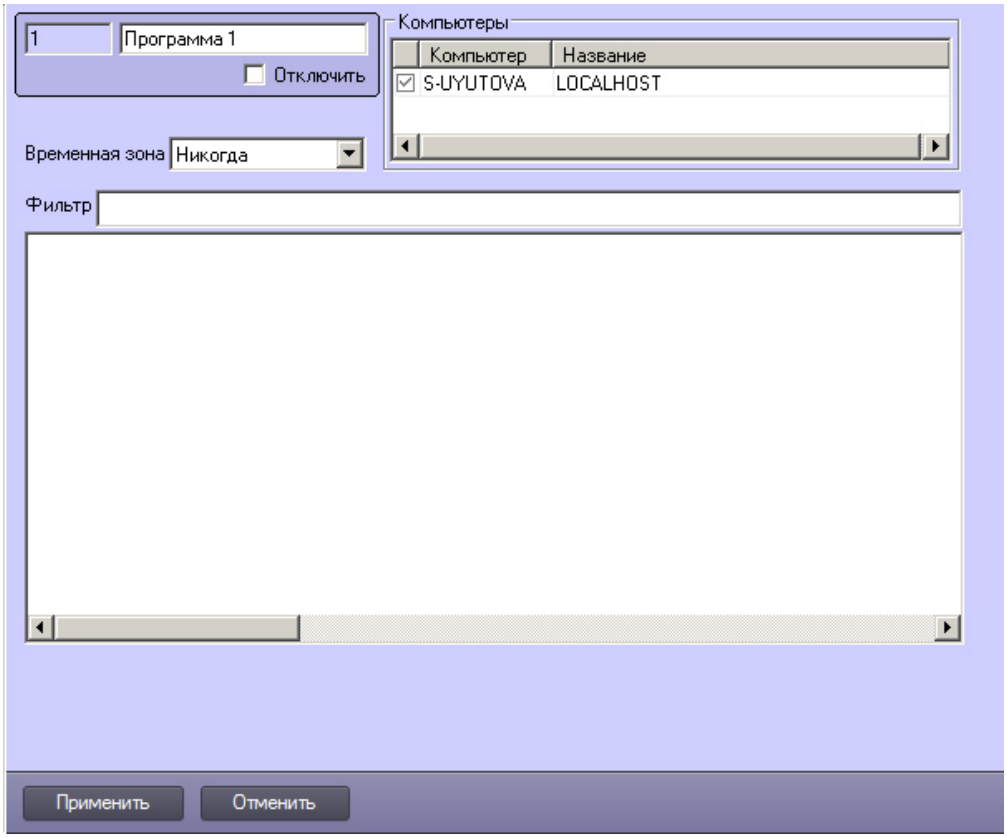
Системный объект **Программа** создается на базе объекта **Программы** на вкладке **Программирование** диалогового окна **Настройка системы**.



Внимание!

Создание большого количества (более 100) объектов **Программа** на одном компьютере может привести к нестабильной работе системы.

Панель настройки системного объекта **Программа** представлена на рисунке:



В панели настройки системного объекта **Программа** указываются временная зона выполнения программы и компьютеры (ядра), на которых требуется выполнять программу.

> **Примечание.**
>
> Для того, чтобы установить флажки напротив всех компьютеров, необходимо выделить ячейку в столбце с флажками и нажать Ctrl+A. Для снятия всех флажков необходимо выделить ячейку и нажать Shift+A.

Для предварительной фильтрации обрабатываемых программой событий следует задать значение в поле **Фильтр**. Формат фильтра – ТИП|ID|СОБЫТИЕ, разделенные точкой с запятой. Например, фильтр CAM|| MD_STOP;CAM||MD_START позволит отфильтровать события **Тревога** и **Конец тревоги** от всех объектов **Камера**.

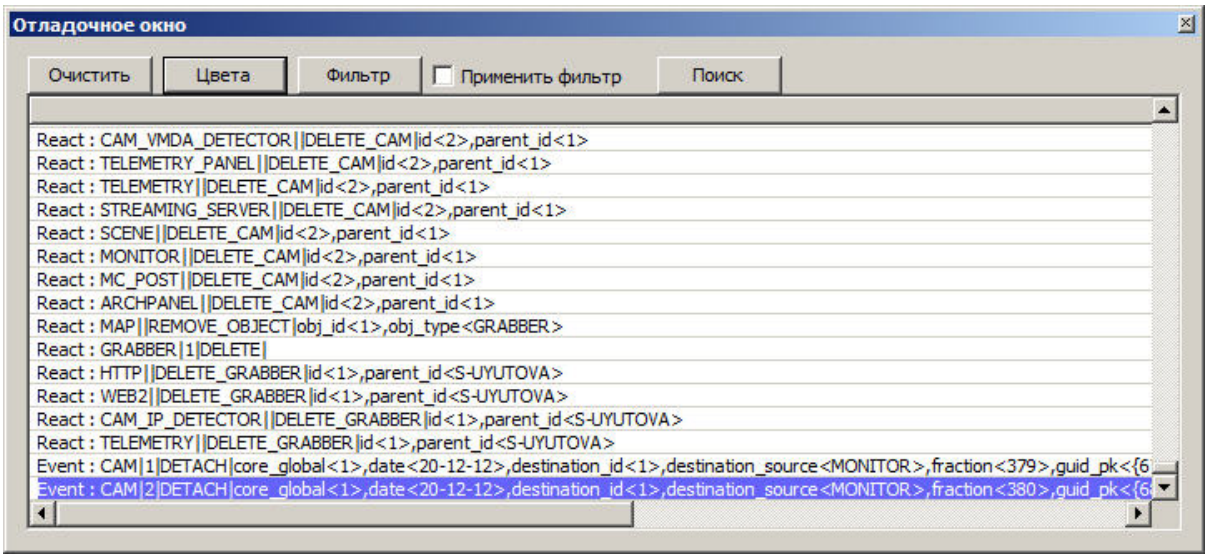
На панели настройки системного объекта **Программа** размещен текстовый редактор для написания и редактирования кода программы.

В текстовом редакторе на панели настроек системного объекта **Программа** есть возможность отмены действия и повтора с помощью горячих клавиш. Для отмены действия нажмите **Alt**+**Backspace**, для повтора – **Ctrl**+**Y**.

6.1.2 Отладочное окно

Отладочное окно программного комплекса *Интеллект* предназначено для просмотра сведений обо всех событиях, регистрируемых в системе.

Вызов **Отладочного окна** выполняется с помощью команды **Отладочное окно** из меню **Выполнить** Главной панели управления. Отладочное окно программного комплекса *Интеллект* отображается в нижней части экрана.



По умолчанию **Отладочное окно** недоступно для вызова. Активирование **Отладочного окна** выполняется с помощью утилиты *tweaki.exe* (см. раздел Отладочное окно документа Руководство по программированию (JScript)).

6.1.3 Синтаксический анализатор

Встроенный синтаксический анализатор позволяет отслеживать правильность написания основных зарегистрированных слов, таких как OnEvent, DoReact, OnTime, Wait, Sleep и др. Эти зарегистрированные слова отмечаются черным цветом в поле текста программы. Следует отметить, что за правильностью написания параметров команд анализатор не следит, и нужно быть особенно внимательным в этих случаях.

```
OnEvent("MACRO","2","RUN")
{[
fn="D:\Intellect\Bmp\Person\1.bmp";

DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<"+fn+">");
DoReact("DIALOG","operator","CLOSE_ALL");
Sleep(500);
DoReact("DIALOG","operator","RUN");
]}

OnEvent("MACRO","3","RUN")
{[
fn="D:\Intellect\Bmp\Person\1.bmp";

DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<"+fn+">");
```

Для изменения размера шрифта используйте сочетания клавиш:

- **CTRL** и **+** для увеличения шрифта

```
OnInit()
{
n1a="0";
n1v="0";
}

OnEvent("OLXA_LINE","1","ACCU_START")
{
n1a="1";
DoReact ("CAM","1","REC");
}
```

- **CTRL** и **-** для уменьшения шрифта

6.1.4 Рекомендуемый порядок написания программ

На странице:

1. Постановка общей задачи.
2. Разбитие задачи на подзадачи.
3. Написание подзадач и их отладка.
4. Поиск и исправление ошибок.

6.1.4.1 Постановка общей задачи

Нужно четко представлять, что должно происходить в системе при определенных событиях. Определить ID устройств, участвующих в генерации событий и действий.

6.1.4.2 Разбитие задачи на подзадачи

Если задача подразумевает обработку нескольких различных событий, то имеет смысл четко представить действия системы на каждое из этих событий. По возможности нужно исключить возможность бесконечного

зацикливания выполнения скриптов, т.е исключить всяческие рекурсивные действия, если конечно они не предусматривают выполнение поставленной задачи.

6.1.4.3 Написание подзадач и их отладка

Наиболее сложным в написании скриптов является написание списка действий с возможным использованием логических и циклических операций. По опыту эта часть программирования наиболее долго отлаживается. Зачастую генерация события, требующая обработки, является не очень удобной, тем более на реальном объекте – например, срабатывание пожарного датчика или движение по камере, достаточно удаленной от места программирования (от сервера с ядром системы). В этом случае рекомендуется на этапе отладки действий генерировать событие вручную, самое удобное – это запуск пустой макрокоманды. После отладки тела скрипта в событие вместо запуска пустой макрокоманды подставляется реальное событие. Кроме того можно проверить и, наоборот, убедиться в правильности написания реального события, не запуская списка действий, вставив вместо списка действий запуск пустой макрокоманды и посмотреть ее выполнение в отладочном окне.

6.1.4.4 Поиск и исправление ошибок

Встроенный синтаксический анализатор на этапе запуска программы проверяет правильность написания названий функций, но не проверяет правильность синтаксиса программы (расстановки ключевых символов: запятых, точек с запятой, вложенность скобок). Чтобы отследить ошибки в программе, если они есть, необходимо активировать режим отладки **Debug 4** (см. Включение и настройка режима отладки программного комплекса Интеллект). В случае наличия синтаксических ошибок на этапе исполнения тела программы отобразится окно **Критические ошибки**, в котором будут перечислены названия функций с неверным синтаксисом и другая отладочная информация.

Примечание

В том случае, если синтаксис программы правильный, но программа не работает или работает с ошибками, рекомендуется переписать программу в виде скрипта на языке JScript (см. Руководство по программированию (JScript)).

6.2 Описание синтаксиса

Скрипт состоит из набора процедур.

Все операторы, выполняемые внутри процедур, формируются в блоки {..}.

Если нужно вставить комментарий, то перед комментарием требуется поставить спецсимволы //.

6.2.1 Описание переменных

Все переменные, используемые в системе – строковые.

Для сравнения строковых переменных и значений используется функция: bool strequal (строка1,строка2). Функция "strequal" возвращает значение, отличное от нуля, если строки равны (см. раздел Описание функций).

Для произведения целочисленных действий используется функция: str(строка1) (см. раздел Описание функций).

6.2.2 Описание процедур

6.2.2.1 Стандартные процедуры

Существуют 3 стандартные процедуры, которые могут быть выполнены при возникновении соответствующего события:

1. **OnInit**() – используется для инициализации переменных (задания первоначальных значений), которые будут в дальнейшем использоваться при выполнении скриптов. Выполняется до старта всех модулей системы. Рекомендуется использовать один вызов процедуры на все существующие скрипты.
   Пример использования:

```
OnInit(){
                flag=1;
                num=8;  //на старте системы будут проинициализированы переменные
}
```

2. **OnTime**(день недели (1-7), день-месяц-год, часы, минуты, секунды) – запуск в определенный момент времени.

```
OnTime(W,D,X,Y,H,C,S)
{
//W - день недели (0 - понедельник, 6 - воскресенье);
//D - дата в формате "число-месяц-год", 16 августа 2001 года это "16-08-01"
//X,Y - зарезервировано
                //H - час
                //C - минуты
                //S - секунды
                // ВЫПОЛНЯЯ СРАВНЕНИЕ С ПАРАМЕТРАМИ, ДАЛЕЕ УКАЗЫВАЕТСЯ ДЕЙСТВИЕ
}
```

   Примеры использования:

```
OnTime(W,"16-08-01",X,Y,"11","11","30")
{
                // помещенный здесь код сработает 16 августа 2001 года в 11 часов 11 минут
 30 секунд
}
```

```
OnTime(W,D,X,Y,"11","11","30")
{
                // помещенный здесь код сработает каждый день в 11 часов 11 минут 30
секунд
}
```

```
OnTime(W,"16-08-01",X,Y,H,C,S)
{
                // помещенный здесь код будет срабатывать 16 августа 2001 года
                // каждую секунду
}
```

```
OnTime(W,"16-08-01",X,Y,"11","11",S)
{
                // помещенный здесь код будет срабатывать 16 августа 2001 года
                // с 11 часов 11 минут по 11 часов 12 минут каждую секунду
}
```

```
OnTime("0",D,X,Y,"21","0","0")
{
                // помещенный здесь код будет срабатывать каждый понедельник
                // в 21 часов 00 минут 00 секунд
}
```

3. **OnEvent**(тип источника, номер, событие) – запуск по определенному событию от объекта системы. Основная процедура при написании скриптов.
   Примеры использования:

```
OnEvent("GRAY","1","ON")
{
                // Выполнится при замыкании луча №1
}
```

```
OnEvent("CAM","12","MD_START")
{
                // Выполнится при срабатывании детектора движения камеры №12
}
```

Каждая процедура, имеющая параметры, может встречаться в коде много раз с различными параметрами. При возникновении события система выполнит те из них, параметры которого совпадут с параметрами возникшего события.

Параметр процедуры может быть определенным или нет. В первом случае его значение берется в кавычки, в последнем случае параметр обозначается латинскими буквами, и процедура будет выполнена для всех событий, для которых его можно определить.

Примеры использования:

```
OnEvent("GRAY","1","ON")       // Выполнится при замыкании луча №1
{
    i=1;
    i=i+1;     //т.к. переменные строковые, то сумма будет равна «11»
    j=1;
    j=str(j+1);           // str – это функция преобразования числа к строке. Внутри функции
 str вначале происходит конвертация всех строковых переменных (в случае их наличия) в
целочисленные, затем происходит сложение чисел, следовательно сумма будет равна «2»
```

```
}
```

```
OnEvent("GRAY",N,"ON")  // Выполнится при замыкании любого луча
{
    if(strequal(N,"3")
    {
       // выполнится если это луч 3
    }
}
```

6.2.2.2 Создание собственных процедур

Все собственные процедуры, описанные в скрипте, должны находиться в том же теле программы и перед процедурами, в которых они вызываются.

```
procedure ProcedureName(список параметров){
    //тело процедуры
}
```

> **Внимание!**
>
> Имена параметров должны состоять из одного символа в верхнем регистре.

Примеры использования:

```
procedure ProcedureName(A,B)

{
    n=A+" "+B;
    //при запуске макроса 1 n=«Макрокоманда 1», при запуске макроса 16 n=«Макрокоманда 16»
}


OnEvent("MACRO",N,"RUN")

{
    a1=N;
    a2="Макрокоманда";
    ProcedureName(a2,a1);
}
```

6.2.3 Описание операторов

Список операторов, используемых для описания действий:

1. **DoReact**(тип объекта, номер, действие[,параметры]) – выполнить действие.
   Пример использования:

```
OnEvent("GRAY","1","ON")
{
                DoReact("GRELE","1","ON");     //при замыкании луча 1 замкнуть реле 1
```

```
}
```

2. **DoCommand**(командная строка) – запуск командной строки.
Пример использования:

```
OnEvent("GRAY","1","ON")
{
                DoCommand("notepad.exe"); //при замыкании луча 1 запустить «Блокнот»
}
```

3. **Wait**(кол-во секунд) – ждать N секунд;
Sleep(кол-во миллисекунд) – ждать N миллисекунд.
Операторы ожидания должны быть выделены в отдельный поток. Отдельный поток выделяется квадратными скобками.
Пример**.** При замыкании Луча №1 Реле №1 будет замыкаться на 5 секунд.

```
OnEvent("GRAY","1","ON")
{
                [
                DoReact("GRELE","1","ON");
                Wait(5);
                DoReact("GRELE","1","OFF");
                ]
}
```

4. Функция проверки состояния объекта:
CheckState(тип объекта, номер, состояние) – результат будет равен 1, если состояние объекта соответствует действительности, иначе 0.
В качестве параметров могут быть выражения. Константные значения берутся в кавычки.
Пример. При замыкании луча №1 проверяется состояние камеры №2 и если состояние «Тревога», то замкнуть реле №1.

```
OnEvent("GRAY","1","ON")
{
                if(CheckState("CAM","2","ALARMED"))
                {
                                DoReact("GRELE","1","ON");
                }
}
```

5. **Условный оператор**:

```
If (выражение)                {
                ... // если результат выражения не 0
                }
                else
                {
                ... // если результат выражения равен 0
                }
```

Часть оператора else {} может отсутствовать.
Пример использования:

```
OnEvent ("MACRO","1","RUN"){
```

```
                x=5;
                if(x>10) {y=2;}   // если "x" больше чем 10, то y=2
                else {y=3;}          //иначе y=3
}
```

6. **Оператор цикла**:

```
For (выражение 1; выражение 2; выражение 3){
                ...
}
```

Выражение 1 выполнится в начале цикла, пока выражение 2 истинно, будет выполняться тело цикла, после каждого выполнения тела цикла будет выполняться выражение 3.
Пример. При замыкании луча №1 реле №1 будет замыкаться и размыкаться с интервалом в 1 секунду и это будет происходить 10 раз.

```
OnEvent ("GRAY","1","ON")
{
                [
                for(i=0;i<10;i=str(i+1))
                {
                  DoReact("GRELE","1","ON");
                  Wait(1);
                  DoReact("GRELE","1","OFF");
                  Wait(1);
                }
                ]
}
```

7. **DoReactGlobal**(тип объекта, номер, состояние) – функция, генерирующая реакции системных объектов. При этом генерируемая реакция рассылается по всем ядрам системы, соединенным по сети.
Пример. При выполнении макрокоманды №1 ставить камеру №1 на охрану.

```
OnEvent("MACRO","1","RUN")
{
    DoReactGlobal("CAM","1","ARM");
}
```

8. **NotifyEventGlobal**(тип объекта, номер, состояние) – функция, генерирующая системные события. При этом генерируемые события рассылаются по всем ядрам системы, соединенным по сети.
Пример. При выполнении макрокоманды №1 генерировать событие «Запись на диск» для камеры №1. Команду отправлять по всем ядрам системы в виде события для регистрации в Протоколе событий.

```
OnEvent("MACRO","1","RUN")
{
    NotifyEventGlobal ("CAM","1","REC");
}
```

Примечание.

Если нет необходимости в рассылке события по всем ядрам системы, можно воспользоваться функцией **NotifyEvent**.

6.2.4 Операции и выражения

В таблице представлены общее описание и примеры использования операций сравнения, арифметических и условных операций.

Оператор	Общее описание, пример использования
Операции сравнения	
>	Оператор сравнения – больше. Пример см. в разделе Описание операторов
<	Оператор сравнения – меньше. Пример см. в разделе Описание операторов
Арифметические операции	

+	Операция сложение. Пример использования:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  x=5;`
	`  y=10;`
	`  i=x+y;  // складывает как строковые т.е. 5+10=510`
	`  e=str(x+y); // складывает как числа 5+10=15`
	`}`

-	Операция вычитание. Пример использования:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  x=5;`
	`  y=10;`
	`  i=x-y;  //  вычитание как числа 5-10=-5`
	`  e=str(x-y); // вычитание как числа 5-10=-5`
	`}`

*	Умножение. Пример использования:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  x=5;`
	`  y=10;`
	`  i=x*y;  // умножает как числа 5*10=50`
	`  e=str(x*y); // умножает как числа 5*10=50`
	`}`

/	Деление. Пример использования:

Оператор	Общее описание, пример использования

	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  x=5;`
	`  y=10;`
	`  i=x/y;  // делит как числа 5/10=0.5`
	`  e=str(x/y); // делит как числа 5/10=0.5`
	`}`

%	Остаток от целочисленного деления. Пример использования.
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  a=1120.0;`
	`  b=100;`
	`  e=a%b;  // остаток от целочисленного деления, т.е. 1100 делится на 100, а 20 – это остаток.`
	`  // если делится без остатка то результат = 0`
	`}`

()	Группа арифметических операций. Пример использования.
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`  x=100/((5*8)/1.028);`
	`}`

Логические операции	

&&	Оператор логическое И. Пример использования:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`a=1;`
	`b=2;`
	`z=3;`
	`if((a<b)&&(b<z))`
	`{`
	`  y=1; //если ложь, то else`
	`}`
	`  else {x=0;}`
	`}`

!	Оператор логического отрицания. Пример использования:
	`OnEvent ("CAM",N,"MD_START")`
	`{`

Оператор	Общее описание, пример использования

	`if(!(strequal(N,"1",)))`
	`{`
	`DoReact("GRELE","1",""ON)`
	`}`
	
	`else`
	`{`
	`DoReact("GRELE","2",""ON)`
	`}`
	`}`

6.2.5 Описание функций

Общее описание и примеры использования математических функций, функций преобразования, форматирования и строковых функций показаны в таблице.

Функции (В квадратных скобках указано количество исполняемых параметров)	Общее описание, пример использования
МАТЕМАТИЧЕСКИЕ	
sin[1]	Тригонометрическая функция расчета синуса угла. Формат: y=sin(x); где y – значение функции, x – аргумент функции (в радианах) Пример: y=sin(1.6) Полученное событие: Event : CORE VAR_CHANGED nt_obj_id<1>,value<0.997495>,name<y>,time<15:26:41>,date<21-09-04>
cos[1]	Тригонометрическая функция расчета косинуса угла. Формат: y=cos(x); где y – значение функции, x – аргумент функции (в радианах) Пример: y=cos(2.2) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.588501>,name<y>,time<16:00:45>,date<21-09-04>
tan[1]	Тригонометрическая функция, возвращает тангенс угла. Формат: y=tan(x); где y – значение функции, x – аргумент функции (в радианах) Пример: y=tan(1) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.557408>,name<y>,time<16:43:45>,date<21-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
asin[1]	Возвращает арксинус заданного числового выражения. Формат: y=asin(x); где y – значение функции (в радианах), x – аргумент Пример: y=asin(0.5) Полученное событие: Event : CORE_VAR_CHANGED int_obj_id<1>,value<0.523599>,name<y>,time<16:46:39>,date<21-09-04>
acos[1]	Возвращает арккосинус заданного числового выражения. Формат: y=acos(x); где y – значение функции (в радианах), x – аргумент Пример: y=acos(0.55) Полученное событие: Event : CORE_VAR_CHANGED int_obj_id<1>,value<0.988432>,name<y>,time<16:46:39>,date<21-09-04>
atan[1]	Возвращает арктангенс заданного числового выражения. Формат: y=atan(x); где: y – значение функции (в радианах), x – аргумент Пример: y=atan(1.2) Полученное событие: Event : Event : CORE_VAR_CHANGED int_obj_id<1>,value<0.876058>,name<y>,time<17:07:09>,date<21-09-04>
sinh[1]	Функция sinh возвращает гиперболический синус значения аргумента. Формат: y=sinh(x); где y – значение функции, x – аргумент функции Пример: y=sinh(0.8) Полученное событие: Event : CORE_VAR_CHANGED int_obj_id<1>,value<0.888106>,name<y>,time<17:12:26>,date<21-09-04>
cosh[1]	Функция cosh возвращает гиперболический косинус значения аргумента. Формат: y=cosh(x); где y – значение функции, x – аргумент функции Пример: y=cosh(0.35) Полученное событие: Event : CORE_VAR_CHANGED int_obj_id<1>,value<0.336376>,name<y>,time<17:25:25>,date<21-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
tanh[1]	Тригонометрическая функция расчета угла. Формат: y=tanh(x); где y – значение функции, x – аргумент функции Пример: y=tanh(0.35) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.419068>,name<y>,time<17:25:25>,date<21-09-04>
exp[1]	Возвращает значение функции e^x, где x - заданное числовое выражение. Формат: y=exp(x); где y – значение функции, x – аргумент Пример: y=exp(1.65) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<5.20698>,name<y>,time<17:39:22>,date<21-09-04>
log[1]	Возвращает натуральный логарифм (по основанию «e») заданного числового выражения. Формат: y=log(x); где: y – значение функции, x – аргумент Пример: y=log(0.65) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.430783>,name<y>,time<17:43:22>,date<21-09-04>
log10[1]	Возвращает десятичный логарифм (по основанию 10) заданного числового выражения. Формат: y=log10(x); где y – значение функции, x – аргумент Пример: y=log10(0.05) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<-1.30103>,name<y>,time<17:46:28>,date<21-09-04>
sqrt[1]	Возвращает квадратный корень из заданного числового выражения. Формат: y=sqrt(x); где y – значение функции, x – аргумент Пример: y=sqrt(9) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<3>,name<y>,time<17:25:25>,date<21-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
abs[1]	Функция abs возвращает абсолютное значение целого аргумента. Формат: y=abs(x); где y – значение функции, x – аргумент Пример: y= abs(-1) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<y>,time<13:39:37>,date<22-09-04>
deg[1]	Тригонометрическая функция расчета угла. Возвращает градусную меру. Формат: y=deg(x); где y – значение функции в градусах, x – значение аргумента в радианах Пример: y=deg(3.14) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<179.908748>,name<y>,time<13:13:51>,date<22-09-04>
rad[1]	Тригонометрическая функция расчета угла. Формат: y=rad(x); где y – значение функции в радианах, x – значение аргумента в градусах Пример: y=rad(180) Полученное событие: Event : CORE VAR_CHANGED value<3.141593>,name<y>,time<15:04:17>,date<17-03-08>
ПРЕОБРАЗОВАНИЕ	
floor[1]	Функция преобразования до целого числа в меньшую сторону. Формат: x= floor(y); где x – значение функции, y – дробное или целое число Пример: x= floor(5.55) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<5>,name<x>,time<20:51:48>,date<21-09-04>
ceil[1]	Функция преобразования до целого числа в большую сторону. Формат: x= ceil (y); где x – значение функции, y – дробное или целое число Пример: x= ceil(5.55) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<x>,time<20:51:48>,date<21-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

str[1]	Функция преобразования числа к строке.
	Формат: x=str(y); где x – значение функции, y – аргумент
	Пример:
	z=(9);
	a=str(z);
	b=sqrt(a);
	Полученные события:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<z>,time<14:27:31>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<a>,time<14:27:31>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<3>,name<b>,time<14:27:31>,date<22-09-04>

atof[1]	Функция преобразования строки в число. Формат: x=atof(y); где x – значение функции, y – аргумент Пример: x="0"; x=str(atof(x)+10); Полученное событие: Event : CORE VAR_CHANGED value<0>,name<x>,time<15:34:44>,date<17-03-08> Event : CORE VAR_CHANGED value<10>,name<x>,time<15:34:44>,date<17-03-08>
val[1]	Функция преобразования строки в число. Формат: x=val(y); где x – значение функции, y – аргумент Пример: x="10"; x=str(val(x)+2); Полученное событие: Event : CORE VAR_CHANGED value<10>,name<x>,time<15:34:44>,date<17-03-08> Event : CORE VAR_CHANGED value<12>,name<x>,time<15:34:44>,date<17-03-08>

int[1]	Преобразование дробного числа в целое (отбросить дробную часть).
	Формат: x=int(y); где x – значение функции, y – аргумент (дробное число для преобразования)
	Пример:
	y=(2.33);
	x=int(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<2.33>,name<y>,time<16:05:28>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<2>,name<x>,time<16:05:28>,date<22-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
long2time[1]	Используется для преобразования заданного количества секунд во время. Формат: x=long2time(y); где x – значение функции(время), y – число в секундах Формат исходной записи (аргумента): <MM> Формат полученной записи: <ЧЧ:ММ:СС> Пример: x=long2time(12345); Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<03:25:45>,name<x>,time<13:53:02>,date<20-09-04>.

time2long[1]	Преобразовать время в количество секунд.
	Формат: x=time2 long(y); где x – значение в секундах, y – время в формате <часы>.<минуты>
	Пример:
	y=(0.15);
	x=time2long(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<0.15>,name<y>,time<19:39:49>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<900>,name<x>,time<19:39:49>,date<22-09-04>

scalar2date[1]	Преобразовать количество дней в дату (количество дней исчисляется с начала нашей эры).
	Формат: x=scalar2date (y); где x – значение(дата), y – количество дней
	Пример:
	y=(731500);
	x=scalar2date(y);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<731500>,name<y>,time<19:57:46>,date<22-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<12-10-03>,name<x>,time<19:57:46>,date<22-09-04>

scalar[1]	Преобразовать дату в количество дней (количество дней исчисляется с начала нашей эры). Формат: x=scalar(y); где x – числовое значение (в сутках), y – дата Формат записи: <ДД.ММ.ГГГГ> Пример: x=scalar(“19.10.2004”) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<10>,value<731873>,owner<WS1>,name<x>,time<15:24:11>, guid_pk<{42E93AF5-4862-485E-AEF6-D14C7BF79C5B}>,date<08-12-09>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

convert_num[1]	Преобразовать число в строку написания этого числа.
	
	Формат: x=convert_num(y); где x – строковое значение числа, y – преобразуемое число
	
	Пример:
	
	y=(24009921);
	
	x=convert_num(y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<24009921>,name<y>,time<12:37:20>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Двадцать четыре миллиона девять тысяч девятсот двадцать один >,name<x>,time<12:37:20>,date<23-09-04>

convert_cur[1]	Преобразовать число (денежную сумму) в строку написания этого числа и добавить руб. и коп.
	
	Формат: x=convert_cur(y); где x – строковое значение денежной суммы, y – число(денежная сумма)
	
	Формат записи: <PP.KK>
	
	Пример:
	
	y=(17999.98);
	
	x=convert_cur(y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.98>,name<y>,time<12:49:30>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Семнадцать тысяч девятсот девяносто девять рублей 98 копеек>,name<x>,time<12:49:30>,date<23-09-04>

ФОРМАТИРОВАНИЯ	

number_frm[2]	Форматирование числа.
	
	Формат: x=number_frm(y,z); где x – значение функции, y-исходное число, z – количество цифр после запятой
	
	Пример:
	
	y=(17999.09998);
	
	x=number_frm(y,3);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.09998>,name<y>,time<14:21:24>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.100>,name<x>,time<14:21:24>,date<23-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

int_frm[2]	Форматирование числа.
	
	Формат: x=int_frm(y,z); где x – значение, y-исходное число, z – количество цифр в числе на выходе
	
	Пример:
	
	y=(17999.99);
	
	x=int_frm(y,10);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<17999.99>,name<y>,time<14:31:46>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<0000017999>,name<x>,time<14:31:46>,date<23-09-04>

currency_std[1]	Форматирование значения числа, представляющего деньги (замена '.' на '-'). Формат: x=currency_std(y); где x – значение функции с измененным форматом, y – число (денежная сумма) Пример: x=currency_std(3.62); Полученное событие: Event : CORE VAR_CHANGED int_obj_id<1>,value<3-62>,name<x>,time<13:40:01>,date<23-09-04>
IsVarExist[1]	Функция проверки заданного параметра в событии. Формат: y=IsVarExist(“x”); где y - значение, x – параметр Если параметр существует, возвращается «1», иначе «0». Пример: p=IsVarExist(“param0”) Полученное событие: Event : CORE VAR_CHANGED int_obj_id<10>,value<0>,owner<WS1>,name<p>,time<12:02:11>, guid_pk<{6A8B5BC9-919C-4098-844A-FBF78FA20820}>,date<14-12-09>
GetObjectIdByParam [3]	Функция, возвращающая первый найденный идентификатор объекта по заданному значению параметра. Id=GetObjectIdByParam ("x","y","z"); где id - возвращаемое значение, x – тип объекта, y – параметр, z – значение параметра Пример: Id=GetObjectIdByParam("CAM","color","0"); Полученное событие: Event : CORE VAR_CHANGED date<28-02-11>,value<2>,int_obj_id<1>,fraction<218>,name<Id>, guid_pk<{F903A28C-3243-E011-901F-6CF049E58698}>,time<15:02:04>,owner<D-IVANOV> * Id=2 (см. value<2>), если функция возвращает пустое значение (value< >), необходимо проверить правильность написания параметров и самой функции.
СТРОКОВЫЕ	

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

strequal[2]	Сравнение строк.
	
	Формат: x=strequal(z,y); где x – значение, z и y – сравниваемые строки
	
	Пример:
	
	z=str(1019);
	
	y=str(1019);
	
	x=strequal(z,y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1019>,name<z>,time<16:51:45>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1019>,name<y>,time<16:51:45>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<x>,time<16:51:45>,date<23-09-04>
	
	* «value<1>» (см. пример выше) – в полученном событии мы получаем либо «value<>» – это означает что сравниваемые строки не совпадают, либо «value<1>» – это значит что сравниваемые строки полностью идентичны друг другу.

strsub[2]	Определение наличия подстроки в строке.
	
	Формат: x=strsub(y,z); где x-значение, y – строка, в которой ведется поиск, z-подстрока
	
	Пример №1:
	
	z=str(888123);
	
	y=str(123);
	
	x=strsub(z,y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<888123>,name<z>,time<16:07:07>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<123>,name<y>,time<16:07:07>,date<23-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<4>,name<x>,time<16:04:34>,date<23-09-04>
	
	Пример №2:
	
	z="67hb8vc56";
	
	y="vc";
	
	x=strsub(z,y);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED value<67hb8vc56>,name<z>,time<12:15:09>,date<18-03-08>
	
	Event : CORE VAR_CHANGED value<vc>,name<y>,time<12:15:09>,date<18-03-08>
	
	Event : CORE VAR_CHANGED value<6>,name<x>,time<12:15:09>,date<18-03-08>
	
	* "value<4>" (см. пример № 1) означает индекс в исходной строке, начиная с которого обнаружено первое вхождение подстроки в эту строку. При отрицательном результате поиска функция возвращает значение value<>.

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**
strempty[1]	Определение пуста ли строка. Формат: x=strempty(y); где x – значение (равно 1 если строка пуста), y – строка Пример: y=""; x=strempty(y); Полученное событие: Event : CORE VAR_CHANGED value< >, name<y>,time<12:27:32>,date<18-03-08> Event : CORE VAR_CHANGED value<1>,name<x>,time<12:27:32>,date<18-03-08> * значение функции value <> означает, что строка не пуста.

straleft[2]	Выравнивание влево.
	Формат: x=straleft(y,z); где x – выровненная строка, y – строка, z – значение выравнивания.
	Пример:
	y=str(123456789);
	x=straleft(y,5);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<123456789>,name<y>,time<18:04:05>,date<23-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<12345>,name<x>,time<18:04:05>,date<23-09-04>
	Примечание. В случае, если число z больше, чем количество символов в строке, функция дополняет исходную строку пробелами справа до тех пор, пока ее длина не станет равна числу z.

strmid[3]	Взять подстроку.
	Формат: x=strmid(y,z,w); где x – строковое значение, y – строка, z – с какой позиции строки, w – длина подстроки
	Пример:
	z=(7);//с какой позиции
	w=(9);//длина
	x=strmid("взять подстроку (1 - строка, 2 - с какой позиции, 3 - длинна)",z,w);
	y=strmid("взять подстроку (1 - строка, 2 - с какой позиции, 3 - длинна)",17,10);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<z>,time<14:18:08>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<9>,name<w>,time<14:18:08>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку>,name<x>,time<14:18:08>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1 – строка>,name<y>,time<14:18:08>,date<24-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

strletf[2]	Взять левую часть строки.
	
	Формат: y=strleft(s,w); где y –строковое значение, s – строка, w – длина (с начала строки)
	
	Пример:
	
	w=(5);//длина
	
	s=("Взять левую часть строки");//строка
	
	y=strleft(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<5>,name<w>,time<14:54:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять левую часть строки>,name<s>,time<14:54:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять>,name<y>,time<14:54:31>,date<24-09-04>

strright[2]	Взять правую часть строки (1 – строка, 2 – длина).
	
	Формат: y=strleft(s,w); где y – строковое значение, s – строка, w – длина (с конца строки)
	
	Пример:
	
	w=(6);//длина
	
	s=("Взять правую часть строки");//строка
	
	y=strright(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:10:36>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Взять правую часть строки>,name<s>,time<15:10:36>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<строки>,name<y>,time<15:10:36>,date<24-09-04>

strnleft[2]	Взять без левой части строки.
	
	Формат: y=strnleft(s,w); где y – строковое значение, s – строка, w – длина левой части, которая будет отсечена
	
	Пример:
	
	w=(6);//длина
	
	s=("взять без левой части строки");//строка
	
	y=strnleft(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:32:38>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без левой части строки>,name<s>,time<15:32:38>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<без левой части строки>,name<y>,time<15:32:38>,date<24-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

srtnright[2]	Взять без правой части строки.
	
	Формат: y=strnright(s,w); где y – строковое значение, s – строка, w – длина правой части, которая будет отсечена.
	
	Пример:
	
	w=(6);//длина
	
	s=("взять без правой части строки");//строка
	
	y=strnright(s,w);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<w>,time<15:44:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без правой части строки>,name<s>,time<15:44:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять без правой части >,name<y>,time<15:44:31>,date<24-09-04>

get_substr[3]	Взять подстроку (1 – строка, 2 – подстрока с которой начать, 3 – подстрока которой завершить, "\r" – конец строки).
	
	Формат: y=get_substr(s,w,x); где y – значение(подстрока), s – строка, w – подстрока с которой начать, x – подстрока которой завершить("\r" – конец строки)
	
	Формат записи: <NN.NN>
	
	Пример:
	
	s=("взять подстроку 1234567890");//строка
	
	w=("по");//подстрока с которой начать
	
	x=("\r");//подстрока которой завершить, "\r" – конец строки
	
	y=get_substr(s,w,x);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять подстроку 1234567890>,name<s>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<по>,name<w>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<\r>,name<x>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку 1234567890>,name<y>,time<16:34:13>,date<24-09-04>
	
	Пример:
	
	s=("взять подстроку 1234567890");//строка
	
	w=("по");//подстрока с которой начать
	
	x=(1);//подстрока которой завершить, "\r" – конец строки
	
	y=get_substr(s,w,x);
	
	Полученное событие:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<взять подстроку 1234567890>,name<s>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<по>,name<w>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<x>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<подстроку >,name<y>,time<16:36:26>,date<24-09-04>

Функции **(В квадратных скобках указано количество исполняемых параметров)**	**Общее описание, пример использования**

strltrim[1]	Убрать пробелы слева.
	Формат: y=strltrim(w); где y – полученное строковое значение, w – строка
	Пример:
	w=(" убрать пробелы слева");//строка
	y=strltrim(w);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value< убрать пробелы слева>,name<w>,time<17:07:49>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<убрать пробелы слева>,name<y>,time<17:07:49>,date<24-09-04>

strrtrim[1]	Убрать пробелы справа.
	Формат: y=strrtrim(w); где y – полученное строковое значение, w – строка
	Пример:
	w=("Убрать пробелы справа ");//строка
	y=strrtrim(w);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Убрать пробелы справа >,name<w>,time<17:18:35>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Убрать пробелы справа>,name<y>,time<17:18:35>,date<24-09-04>

stratrim[1]	Убрать пробелы с обеих сторон.
	Формат: y=stratrim(w); где y – полученное строковое значение, w – строка
	Пример:
	w=(" убрать пробелы с обеих сторон ");//строка
	y=stratrim(w);
	Полученное событие:
	Event : CORE VAR_CHANGED int_obj_id<1>,value< убрать пробелы с обеих сторон >,name<w>,time<17:27:44>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<убрать пробелы с обеих сторон>,name<y>,time<17:27:44>,date<24-09-04>

Примечание.

Функции date<ДД-ММ-ГГ> и time<ЧЧ:ММ:СС> возвращают текущие дату и врем я соответственно. Функция pi<3,14159265358979323846264338327 95> возвращает значение числа π.

6.3 Примеры скриптов на встроенном языке

6.3.1 Пример с BacNet

BACNET BacNet

Формат процедуры событий для объекта **BacNet**:

```
OnEvent("BACNET","_id_","_событие_")
```

Формат оператора для описания действий с объектом **BacNet**:

```
DoReact("BACNET","_id_","_команда_" [,"_параметры_"]);
```

6.3.2 Пример с Пользователем

Формат процедуры событий для объекта **Пользователь**:

```
OnEvent("PERSON","_id_","_событие_")
```

6.3.3 Пример с Ядром

CORE Ядро

Запуск процедуры происходит при возникновении соответствующего события. Формат процедуры событий для объекта **Ядро**:

```
OnEvent("CORE","_id_","_событие_")
```

Пример. При появлении лица в кадре выводить на Монитор 2 видеоизображение с соответствующей камеры. При исчезновении лица убирать с Монитора 2 видеоизображение с соответствующей камеры.

```
OnEvent("CORE",N,"DO_REACT")
{
    if (strequal(action,"SET_MARKRECT"))
    {
        DoReact("MONITOR","2","ADD_SHOW","cam<"+param5_val+">");
    }
    if (strequal(action,"DEL_MARKRECT"))
    {
        [
        Wait(2);
        DoReact("MONITOR","2","REMOVE","cam<"+param0_val+">");
        ]
    }
}
```

6.3.4 Примеры с Архивом и Внешним хранилищем

ARCH Долговременный архив
IPSTORAGE Внешнее хранилище

6.3.4.1 Форматы

Формат процедуры событий для объекта **Долговременный архив**:

```
OnEvent("ARCH","_id_","_событие_")
```

Формат оператора для описания действий с **Внешним хранилищем**:

```
DoReact("IPSTORAGE","_id_","_команда_" [,"_параметры_"]);
```

6.3.4.2 Примеры

Пример для объекта **Долговременный архив**. Если не производится архивация через Долговременный архив 1, разослать соответствующее событие по всем ядрам системы.

```
OnEvent("ARCH","1","INACTIVE")
{
NotifyEventGlobal ("ARCH","1","INACTIVE");
}
```

Пример для объекта **Внешнее хранилище**. По макрокоманде 10 выполнить импорт архива из внешнего хранилища камеры 45 за период с 11-01-19 16:00:55 по 11-01-19 17:00:55.

```
OnEvent("MACRO","10","RUN")
{
    DoReact("IPSTORAGE", "1", "IMPORT", "cam<45>,datetime_from<11-01-19
16:00:55>,datetime_to<11-01-19 17:00:55>");
}
```

6.3.5 Примеры с Аудио

PLAYER Аудиопроигрыватель
OLXA_LINE Микрофон

6.3.5.1 **Форматы**

Формат оператора для описания действий с **Аудиопроигрывателем**:

```
DoReact("PLAYER","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для **Микрофона**:

```
OnEvent("OLXA_LINE", "_id_","_событие_")
```

Формат оператора для описания действий с **Микрофоном**:

```
DoReact("OLXA_LINE","_id_","_команда_" [,"_параметры_"]);
```

Функция проверки состояния объекта **Микрофон**:

```
CheckState("OLXA_LINE","номер","состояние")
```

6.3.5.2 Примеры

Пример использования событий и реакций объекта **Аудиопроигрыватель**:

1. Необходимо проигрывать звуковой файл при остановке записи видеокамеры:

```
OnEvent("CAM",N,"REC_STOP")
{
    DoReact("PLAYER","1","PLAY_WAV","file<C:\Program Files
(x86)\Intellect\Wav\cam_alarm_"+N+".wav>,from_macro<1>");
}
```

2. Необходимо завершать проигрывание файла при начале записи видеокамеры:

```
OnEvent("CAM",N,"REC")
{
    DoReact("PLAYER","1","STOP_WAV");
}
```

3. Проигрывание звукового файла от прихода одного события до прихода другого события (в данном примере это запуск макрокоманд).
   Звуковой файл должен длиться не больше количества секунд, которое указано в операторе Wait.

```
OnEvent("MACRO","1","RUN")

{
    flag=1;
    [
    for(i=1;flag;i=1)
    {
        DoReact("PLAYER","1","PLAY_WAV","file<C:\Program
Files\Intellect\Wav\cam_alarm_1.wav>");
        Wait(3);
    }
    ]
}

OnEvent("MACRO","8","RUN")
{
    flag=0;
}
```

Примеры использования событий и реакций объекта **Микрофон**:

1. Необходимо включить первый микрофон на запись при включении акустопуска.

```
OnEvent("OLXA_LINE","1","accu_start") //включение акустопуска
{
    DoReact("OLXA_LINE","1","ARM"); //включение микрофона на запись
}
```

2. Необходимо установить минимальную компрессию на микрофоне при выключении записи аудиосигнала.

```
OnEvent("OLXA_LINE","1","DISARM") //отключение записи с микрофона
{
    DoReact("OLXA_LINE","1","SETUP","compression<5>"); //установлена минимальная
компрессия
}
```

3. Микрофон (OLXA_LINE) пишется не синхронно с камерой. По умолчанию микрофон не стоит на охране. Необходимо писать звук как по акустопуску, так и по детекции от камеры. На сработку акустопуска (ACCU_START) и детектора движения (MD_START) включается принудительная запись звука и увеличивается на единицу переменная flag. При окончании акустопуска и детекции движения переменная flag уменьшается на единицу и запись звука останавливается, только если она равна нулю, т.е. нет ни акустопуска, ни движения.

```
OnInit()
{
    flag=0;
}

OnEvent("CAM","3","MD_START")
{
    flag=str(flag+1);
    DoReact("OLXA_LINE","1","RECORD_START");
}

OnEvent("OLXA_LINE","1","ACCU_START")
{
    flag=str(flag+1);
    DoReact("OLXA_LINE","1","RECORD_START");
}

OnEvent("OLXA_LINE","1","ACCU_STOP")
{
    flag=str(flag-1);
    if (!(flag))
    {
        DoReact("OLXA_LINE","1","RECORD_STOP");
    }
}

OnEvent("CAM","3","MD_STOP")
{
    flag=str(flag-1);
    if (!(flag))
    {
        DoReact("OLXA_LINE","1","RECORD_STOP");
    }
}
```

6.3.6 Примеры с Видеошлюзом

GATE Видеошлюз

Формат процедуры событий для объекта **Видеошлюз**:

```
OnEvent("GATE","id","событие")
```

Пример. При падении темпа ввода на шлюзе 1 разослать соответствующее событие по всем ядрам системы.

```
OnEvent("GATE ","1"," GATE_LOW_FPS ")
{
    NotifyEventGlobal ("GATE ","1"," GATE_LOW_FPS ");
}
```

Формат оператора для действий с **Видеошлюзом**:

```
DoReact("GATE","id","команда"[,"параметры"]);
```

6.3.7 Примеры с Детекторами

CAM_VMDA_DETECTOR Детектор VMDA
CAM_FACECAPTURE Детектор лиц
CAM_IP_DETECTOR Детектор встроенный

6.3.7.1 Форматы

Формат процедуры событий для **Детектора VMDA**:

```
OnEvent("CAM_VMDA_DETECTOR","_id_","_событие_")
```

Формат оператора для описания действий с **Детектором VMDA**:

```
DoReact("CAM_VMDA_DETECTOR","_id_","_команда_");
```

Формат событий для объекта **Детектор лиц**:

```
OnEvent("CAM_FACECAPTURE","_id_","_событие_")
```

Формат оператора для описания действий с **Детектором лиц**:

```
DoReact("CAM_FACECAPTURE","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Детектор встроенный**:

```
OnEvent("CAM_IP_DETECTOR","_id_","_событие_")
```

6.3.7.2 Пример

Пример использования событий и реакций объекта **Детектор VMDA**:

При выполнении Макрокоманды 1 поставить на охрану Детектор VMDA 2:

```
OnEvent ("MACRO","1","RUN")
{
DoReact("CAM_VMDA_DETECTOR","2","ARM");
}
```

6.3.8 Примеры с Камерами и Монитором видеонаблюдения

GRABBER Устройство видеоввода
MACRO Макрокоманда
CAM Камера
MONITOR Монитор видеонаблюдения

6.3.8.1 Форматы и функции

Формат процедуры событий для **Устройства видеоввода**:

```
OnEvent("GRABBER","_id_","_событие_")
```

Формат оператора для описания действий с **Устройством видеоввода**:

```
DoReact("GRABBER","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Камера**:

```
OnEvent("CAM","_id_","_событие_")
```

Формат оператора для описания действий с **Камерой**:

```
DoReact("CAM","_id_","_команда_" [,"_параметры_"]);
```

Функция проверки состояния объекта **Камера**:

```
CheckState("CAM","номер","состояние")
```

Формат процедуры событий для объекта **Монитор**:

```
OnEvent("MONITOR","_id_","_событие_")
```

Формат оператора для действий с **Монитором**:

```
DoReact("MONITOR","_id_","_команда_"[,"_параметры_"]);
```

6.3.8.2 Примеры

Примеры использования событий и реакций объекта **Устройство видеоввода**:

1. Необходимо установить для первого устройства видеоввода первый канал, максимальную скорость оцифровки, разрешение полукадр и формат PAL при запуске первой макрокоманды.

```
OnEvent("MACRO","1","RUN") //запуск макрокоманды 1
{
    DoReact("GRABBER","1", "SETUP", "chan<1>,mode<0>,resolution<1>,format<PAL>");
    //установка для первой платы видеоввода канал – 1, скорость оцифровки – максимальная,
разрешение – полукадр, формат – PAL
}
```

2. Необходимо при запуске третьей макрокоманды установить диски D:\ и F:\ для записи видеоархива.

```
OnEvent("MACRO","3","RUN") //запуск макрокоманды 3
{
    DoReact("GRABBER","1","SET_DRIVES","drives<D:\,F:\>"); //запись видеоархива на диски
 D:\ и F:\
}
```

3. Необходимо вывести первую видеокамеру на первый аналоговый выход платы и отключить первые аналоговые выходы первой и второй плат при ошибке подключения ко второй плате видеоввода.

```
OnEvent("GRABBER","2"," UPS_FATAL_ERROR") //ошибка подключения к плате видеоввода 2
{
    DoReact("CAM","1","MUX1"); //вывод видеокамеры 1 на 1-ый аналоговый вывод платы
    Wait(5);
    DoReact("GRABBER","1","MUX1_OFF"); //отключение 1-го аналогового выхода первой платы
    DoReact("GRABBER","2","MUX1_OFF"); //отключение 1-го аналогового выхода второй платы
}
```

Примечание.

Если аналоговые выходы двух и более плат соединяются параллельно, и видеокамера 1, например, принадлежит первому грабберу, а видеокамера 2 – второму, то при вызове команды «DoReact("CAM","1","MUX1");» необходимо сначала вызвать команду «DoReact("GRABBER","2","MUX1_OFF");» и, соответственно, при вызове команды «DoReact("CAM","2","MUX1");» необходимо сначала вызвать команду «DoReact("GRABBER","1","MUX1_OFF");». Иначе произойдет наслоение сигналов.

4. Необходимо отключить второй аналоговый выход платы видеоввода при восстановлении питания от сети.

```
OnEvent("GRABBER","1","UPS_ONLINE") //восстановление питания от сети
{
    DoReact("GRABBER","1","MUX2_OFF"); //отключение аналогового выхода 2
}
```

Примеры использования событий и реакций объекта **Камера**:

1. При постановке первой камеры на охрану выполнить перевод камеры в цветной режим и начать запись с нее.

```
OnEvent("CAM","1","ARM") //первая видеокамера поставлена на охрану
{
    DoReact("CAM","1","SETUP","color<1>"); //установка цветного режима видеокамеры
    DoReact("CAM","1","REC"); //запись с первой видеокамеры
}
```

2. Необходимо поставить на охрану первую видеокамеру при отключении пятой видеокамеры.

```
OnEvent("CAM","5","DETACH") //пятая видеокамера отключена
{
    DoReact("CAM","1","ARM"); //первая видеокамера поставлена на охрану
}
```

3. Необходимо использовать половину ресурсов при записи у первой видеокамеры (то есть, если в системе через первую плату видеоввода подключено 4 видеокамеры, то первая будет записывать со скоростью 6 кадров/сек, а остальные три – по 2-2,5 кадра/сек.), если она находится в тревожном состоянии.

```
OnEvent("CAM","1","MD_START") //первая видеокамера находится в тревожном состоянии
{
    DoReact("CAM","1","SETUP","rec_priority<2>"); //использование половины ресурсов при
записи
}
```

4. Необходимо установить максимальную компрессию синхронно с четвертым микрофоном звуковой платы на первой видеокамере при записи на диск видео с первой видеокамеры.

```
OnEvent("CAM","1","REC") //первая видеокамера ведет запись на диск
{
    DoReact("CAM", "1", "SETUP", "compression<5>, audio_type<OLXA_LINE>, audio_id<4>"); //
первая видеокамера, максимальная компрессия,
синхронно с четвертым микрофоном звуковой платы
}
```

5. Необходимо начать запись с первой видеокамеры с минимальным качеством в черно-белом режиме, когда она выйдет из состояния тревоги.

```
OnEvent("CAM","1","MD_STOP") //первая видеокамера перестала находиться в тревожном
состоянии
{
    value = 5;
    DoReact("CAM", "1", "SETUP", "compression<" + value + ">,color<0>");
    //начать запись первой видеокамеры с минимальным качеством в ч/б режиме
}
```

6. Необходимо начать запись с первой видеокамеры в режиме «откат», когда она снята с охраны.

```
OnEvent("CAM","1","DISARM")  //первая видеокамера снята с охраны
{
    DoReact("CAM","1","REC","rollback<1>");  //начать запись с первой видеокамеры в режиме
 «откат»
}
```

7. Установить новые параметры видеоканала при подключении первой видеокамеры.

```
OnEvent("CAM","1","ATTACH") //подключена первая видеокамера
{
    VIDEO_CANAL_ID = GETOBJECTPARAM("CAM","1","PARENT_ID"); //определяем идентификатор
видеоканала, которому принадлежит первая видеокамера
    DoReact("GRABBER",VIDEO_CANAL_ID,"SETUP","chan<0>,mode<0>,resolution<1>,format<pal>"); /
/устанавливаем новые параметры видеоканала
}
```

8. По макрокоманде 2 запустить автопанорамирование на камере 1.

```
OnEvent ("MACRO","2","RUN")
{
    DoReact("CAM","1","CRUISE_START","cruise_id<1>,action<CRUISE_START>,cam_id<1>");
}
```

9. Есть определенное количество камер (num). Необходимо проверить работу детектора движения по всем камерам (можно использовать для проверки работоспособности датчиков охраны).
Для решения задачи используется эмуляция линейного символьного массива (строка), т.е. заполняется массив символов (в примере это символ «N»). Далее при сработке детектора движения по камере меняется соответствующий (идентификатору камеры) элемент массива (меняется на "Y"). Таким образом, на выходе образуется символьный массив из «N» (камера не сработала) и «Y» (камера сработала). Подсчитывается количество сработок и выдается сообщение об общем количестве камер и количество камер, у которых сработал детектор. Старт проверки по Макрокоманде 1. Остановка по Макрокоманде 2.

```
OnInit()
{
    run=0;
}

OnEvent("MACRO","1","RUN")
{
    run=1; flag=""; num=8;
    for(i=1;i<str(num+1);i=str(i+1))
    {
        DoReact("CAM",i,"DISARM");
        DoReact("CAM",i,"REC_STOP");
        DoReact("CAM",i,"ARM");
        flag=flag+"N";
        if(i<num) {flag=flag+"|";}
    }
}

OnEvent("CAM",N,"MD_START")
{
    if(run)
    {
        nn=str((N*2)-1);
        flag=strleft(flag,str(nn-1))+"Y"+strright(flag,str(((num*2)-1)-nn));
    }
}

OnEvent("MACRO","2","RUN")
{
    run=0; fin=0;
    for(i=1;i<str(num+1);i=str(i+1))
    {
        tmp=extract_substr(flag,"|",str(i-1));
        if(strequal(tmp,"Y")) {fin=str(fin+1);}
        DoReact("CAM",i,"DISARM");
    }
    tmp="Всего:"+str(num)+" Сработало:"+str(fin);
    rez=MessageBox("",tmp,0);
}
```

10. При возникновении тревоги по камере 1 накладывать титры на видеоизображение с данной камеры. При окончании тревоги накладывать титры об окончании тревоги.

```
OnEvent("CAM","1","MD_START")
{
    DoReact("CAM","1","CLEAR_SUBTITLES","title_id<1>"); //удалить все титры с
видеоизображения
    DoReact("CAM","1","ADD_SUBTITLES","command<Камера 1 Тревога " + time +
"\r>,page<BEGIN>,title_id<1>");
    //параметр time позволяет включить в титры время регистрации события
}

OnEvent("CAM","1","MD_STOP")
{
    DoReact("CAM","1","ADD_SUBTITLES","command<Камера 1 Конец тревоги " + time +
"\r>,page<END>,title_id<1>");
}
```

> **Примечание**
>
> При использовании параметров page<BEGIN> и page<END> будут заполняться соответствующие поля в базе титров, что даст возможность производить поиск данных с помощью интерфейсного объекта **Поиск по титрам**.

Примеры использования событий и реакций объекта **Монитор**:

1. Необходимо при запуске первой макрокоманды проиграть запись с видеокамеры 1 на мониторе 4 с указанными датой и временем.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("MONITOR","4","ARCH_FRAME_TIME","cam<1>,date<"+date+">,time<11:00:00>");
    DoReact("MONITOR","4","KEY_PRESSED","key<PLAY>");
}
```

2. Необходимо при печати кадра с первой видеокамеры перейти в режим просмотра видеоархива на первой видеокамере монитора 4, и перейти на 10 кадров далее, начиная с фрагмента указанной даты и времени.

```
OnEvent("CAM", "1", "PRINT")
{
    DoReact("MONITOR","4","ARCH_FRAME_TIME","cam<1>,date<"+date+">,time <11:00:00>");
    for(i=0;i<10;i=i+1)
    {
        DoReact ("MONITOR","4","KEY_PRESSED","key<FF>");
    }
}
```

3. Необходимо приблизить видеоизображение на экране монитора, если видеокамера находится в состоянии тревоги, и вернуть в исходное состояние при ее окончании.

```
OnEvent("CAM","1","MD_START")
{
    DoReact("MONITOR","1","KEY_PRESSED","key<ZOOM_IN>");
}
```

```
OnEvent("CAM", "1", "MD_STOP");
{
    DoReact("MONITOR","1","KEY_PRESSED","key<ZOOM_OUT>");
}
```

4. Необходимо вывести на экран монитора раскладку под номером 1 при срабатывании макрокоманды.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("MONITOR","1","KEY_PRESSED","key<SELECT_LAYOUT>,number<1>");
}
```

5. Команда запуска экспорта видео с Камеры 1 в Мониторе 1, начиная с момента времени 24-10-14 17:10:38 и заканчивая 24-10-14 17:10:50, в файл c:\aaa.avi.
   Примеры запуска экспорта тремя способами: через IIDK (порт 900 и порт 1030) и через скрипт.
   a. **IIDK (порт 900)**
      MONITOR|1|START_AVI_EXPORT|start<24-10-14 17:10:38>,finish<24-10-14 17:10:50>,avi_path<c:\aaa.avi>,cam<1>
   b. **IIDK (порт 1030)**
      CORE||DO_REACT|
      source_type<MONITOR>,source_id<1>,action<START_AVI_EXPORT>,params<4>,param0_name<avi_path>,
      param0_val<c:\aaa.avi>,param1_name<cam>,param1_val<1>,param2_name<finish>,param2_val<24-10-14 17:10:50>,
      param3_name<start>,param3_val<24-10-14 17:10:38>
   c. **Скрипт** (запуск по Макрокоманде 1)

```
OnEvent("MACRO","1","RUN")
{
    DoReact("CORE","","DO_REACT","source_type<MONITOR>,source_id<1>,action<START_AVI
_EXPORT>,params<4>,
    param0_name<avi_path>,param0_val<c:
\aaa.avi>,param1_name<cam>,param1_val<1>,param2_name<finish>,
     param2_val<24-10-14 17:10:50>,param3_name<start>,param3_val<24-10-14
17:10:38");
}
```

6. По макрокоманде 1 включать управление телеметрией при помощи мыши на камере 4, выведенной на монитор 10, по макрокоманде 2 отключать.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("MONITOR","10","CONTROL_TELEMETRY","cam<4>,on<1>");
}

OnEvent("MACRO","2","RUN")
{
    DoReact("MONITOR","10","CONTROL_TELEMETRY","cam<4>,on<0>");
}
```

7. Выводить активную камеру на аналоговый монитор.

```
OnEvent ("MONITOR","1","ACTIVATE_CAM")

{
```

```
    DoReact ("CAM",cam,"MUX1");
}
```

8. Выводить тревожную камеру в режим однократера.

```
OnEvent ("CAM",N,"MD_START")
{
    DoReact ("MONITOR","1","ACTIVATE_CAM","cam<"+N+">");
    DoReact ("MONITOR","1","KEY_PRESSED","key<SCREEN.1>");
}
```

9. Тревожный монитор, на котором всегда остается видео от последней тревожной камеры.

```
OnInit()
{
    counter=0;
}

OnEvent("CAM",T,"MD_START")
{
    if(strequal(counter,"0"))
    {
        DoReact("MONITOR","2","REMOVE_ALL");
        DoReact("MONITOR","2","ADD_SHOW","cam<"+T+">");
    }
    counter=str(counter+1);
}


OnEvent("CAM",M,"MD_STOP")
{
    counter=str(counter-1);
    if(strequal(counter,"0"))
    {
        DoReact("MONITOR","2","ADD_SHOW","cam<"+M+">");
    }
}
```

6.3.9 Примеры с Картами

MAP Карта

Формат процедуры событий для **Карты**:

```
OnEvent("MAP", "_id_","_событие_" [,"_параметры_"])
```

Формат оператора для описания действий с **Картой**:

```
DoReact("MAP","_id_","_команда_" [,"_параметры_"]);
```

Пример. Скрыть Камеру 10 на Карте 1 по Макрокоманде 10.

```
OnEvent("MACRO","10","RUN")
```

```
{
    DoReact("MAP","1","HIDE_OBJECT","objtype<CAM>,objid<10>,hide<1>");
}
```

6.3.10 Примеры с Компьютером и Экраном

SLAVE Компьютер
DISPLAY Экран

6.3.10.1 Форматы

Формат процедуры событий для объекта **Компьютер**:

```
OnEvent("SLAVE","_id_","_событие_")
```

Формат оператора для описания действий с объектом **Компьютер**:

```
DoReact("SLAVE","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Экран**:

```
OnEvent("DISPLAY","_id_","_событие_")
```

Формат оператора для описания действий с **Экраном**:

```
DoReact("DISPLAY","_id_","_команда_" [,"_параметры_"]);
```

6.3.10.2 Примеры

Примеры использования событий и реакций объекта **Компьютер:**

1. При отсутствии диска для записи архива остановить запись с камеры 2.

   ```
   OnEvent("SLAVE","1"," NO_DISC")
   {
       DoReact("CAM","2"," REC_STOP");
   }
   ```

2. По Макрокоманде 1 получить глубину архива по камере 1.

   ```
   OnEvent ("MACRO","1","RUN"){
      DoReact ("SLAVE","WS3","GET_DEPTH","cam<1>");
   }
   ```

   В результате в отладочном окне будет отображена следующая строка:

   ```
   Event : SLAVE|WS3|ARCHIVE_DEPTH|
   cam<1>,core_global<1>,date<11-07-13>,depth<42>,destination_id<1>,destination_source<PROGRA
   ```

```
M>,fraction<970>,guid_pk<{003DFC83-0CEA-E211-A437-0017C401D5C2}
>,owner<WS3>,param0<01:18>,slave_id<WS3>,time<13:30:33>
```

Кроме того, в Протоколе событий будет отображено событие **Глубина архива**, а в поле **Дополнительная информация** будет указана глубина архива в формате Дни:Часы. Данная информация также отображается в отладочном окне в параметре события **param0<>**.

Пример использования событий и реакций объекта **Экран**:

1. При активировании первой временной зоны отобразить первый экран на компьютере CLIENT.

```
OnEvent("TIME_ZONE","1","ACTIVATE")
{
    DoReact("DISPLAY","1","ACTIVATE","macro_slave_id<CLIENT>");
}
```

2. Есть 2 экрана, первый отображает виртуальный монитор с камерами, второй отображает объект Карта с датчиками ОПС Болид. При сработке тревоги по камере показывается Экран 1, при срабатывании тревоги от датчика показывается Экран 2, но только на компьютере CLIENT.

```
OnEvent("CAM",N,"MD_START")
{
    DoReact("DISPLAY","2","DEACTIVATE","macro_slave_id<CLIENT>");
    DoReact("DISPLAY","1","ACTIVATE","macro_slave_id<CLIENT>");
}

OnEvent("BOLID_ZONE",M,"ALARM")
{
    DoReact("DISPLAY","1","DEACTIVATE","macro_slave_id<CLIENT>");
    DoReact("DISPLAY","2","ACTIVATE","macro_slave_id<CLIENT>");
}
```

6.3.11 Примеры с Макрокомандами и Временными зонами

MACRO Макрокоманда
TIME_ZONE Временная зона

6.3.11.1 Форматы и функции

Формат процедуры событий для объекта **Макрокоманда**:

```
OnEvent("MACRO","_id_","_событие_")
```

Формат оператора для описания действий с **Макрокомандами**:

```
DoReact("MACRO","_id_","_команда_" ["_параметры_"]);
```

Функция проверки состояния объекта **Макрокоманда**:

```
CheckState ("MACRO","номер","состояние")
```

Формат процедуры событий для объекта **Временная зона**:

```
OnEvent("TIME_ZONE","_id_","_событие_")
```

Формат оператора для описания действий с **Временной зоной**:

```
DoReact("TIME_ZONE","_id_","_команда_" ,"_параметры_"]);
```

Функция проверки состояния объекта **Временная зона**:

```
CheckState ("TIME_ZONE","номер","состояние")
```

6.3.11.2 Примеры

Примеры использования событий и реакций объекта **Макрокоманда**:

1. Необходимо записать текущее положение видеокамеры в 1 пресет при выполнении макрокоманды 1.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("TELEMETRY","1","SET_PRESET","TEL_PRIOR<1>");
}
```

2. Необходимо выполнить макрокоманду 2, если камера 1 поставлена на охрану.

```
OnEvent("CAM","1","ARM")
{
    DoReact("MACRO", "2", "RUN");
}
```

3. Запускать и останавливать патрулирование поворотного устройства по макрокомандам.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("TELEMETRY","1.1","PATROL_PLAY","tel_prior<1>");
}

OnEvent("MACRO","2","RUN")
{
    DoReact("TELEMETRY","1.1","STOP","tel_prior<1>");
}
```

4. Пример бесконечного цикла и выхода из него. Старт цикла по макрокоманде 1, остановка по макрокоманде 2.

```
OnEvent("MACRO","1","RUN") //при запуске макрокоманды 1
{
    //квадратные скобки нужны для выделения оператора ожидания в отдельный поток
    [
    flag=1;
    for(a=1;flag<2;a=1) //оператор цикла
    {
        Sleep(500); //оператор ожидания создает паузу в 500 миллисекунд
```

```
        ff="!!!!!!!!!!!!!!!!!!";
    }
    ]
}

OnEvent("MACRO","2","RUN") //при запуске макрокоманды 2
{
    flag=2;
}
```

Пример использования событий и реакций объекта **Временная зона**:

1. При активировании первой временной зоны вывести на монитор видеоизображение с камеры 1.

```
OnEvent("TIME_ZONE","1","ACTIVATE")
{
    DoReact ("CAM", "1", "ACTIVATE", "MONITOR<1>");
}
```

6.3.12 Примеры с Окном запроса оператора и SIP-терминалом

DIALOG Окно запроса оператора
SIP_TERMINAL SIP-терминал

6.3.12.1 Форматы

Формат оператора для описания действий с **Окном запроса оператора**:

```
DoReact("DIALOG","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **SIP-терминал**:

```
OnEvent("SIP_TERMINAL","_id_","_событие_")
```

Формат оператора для описания действий с **SIP-терминалом**:

```
DoReact("SIP_TERMINAL","_id_","_команда_" [,"_параметры_"]);
```

6.3.12.2 Примеры

Примеры использования реакций объекта **Окно запроса оператора**:

1. Необходимо по макрокоманде с номером 1 устанавливать координаты верхнего левого угла окна запроса оператора (поворотной видеокамеры PANASONIC-850) в центре экрана, запрещать его перемещение и выводить его на экран.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("DIALOG","PANASONIC-850","SETUP","x<50>,y<50>,allow_move<0>");
    DoReact("DIALOG","PANASONIC-850","RUN");
```

```
}
```

2. Необходимо закрывать окно запроса оператора по макрокоманде с номером 2.

```
OnEvent("MACRO","2","RUN")
{
    DoReact("DIALOG","PANASONIC-850","CLOSE");
}
```

6.3.13 Примеры с Поворотными устройствами (PTZ) и Устройствами управления

TELEMETRY Поворотное устройство
TELEMETRY_EXT Пульт управления
IPJOYSTICK Устройство управления

6.3.13.1 Форматы

Формат процедуры событий для объекта **Поворотное устройство**:

```
OnEvent("TELEMETRY ","_id_","_событие_")
```

Формат оператора для описания действий с **Поворотными устройствами**:

```
DoReact("TELEMETRY ","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Пульт управления**:

```
OnEvent("TELEMETRY_EXT ","_id_","_событие_")
```

Формат оператора для описания действий с **Пультом управления**:

```
DoReact("TELEMETRY_EXT","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Устройство управления**:

```
OnEvent("JOYSTICK","_id_","_событие_")
```

6.3.13.2 Примеры

Примеры использования реакций объекта **Поворотное устройство**:

1. Необходимо установить автофокусирование, когда видеокамеру 1 ставят на охрану.

```
OnEvent("CAM","1","ARM")
{
    DoReact("TELEMETRY","1", "AUTOFOCUS_ON");
}
```

2. Необходимо повернуть видеокамеру в положение, заданное в первом пресете, при включении реле.

```
OnEvent("GRELE","1","ON")
{
    telemetry_id= GetObjectParam("CAM","1","parent_id");
    DoReact("TELEMETRY","telemetry_id","SETUP","GO_preset<1>");
}
```

3. Записать маршрут патрулирования для Камеры 1, соответствующей Поворотному устройству 1.1. Маршрут состоит из двух точек, таких, что для перехода из точки 1 в точку 2 необходимо поворачивать камеру влево со скоростью 6 в течение 2 секунд. Патрулирование должно осуществляться со скоростью 10. Время нахождения в каждой точке маршрута – 25 секунд. Предполагается, что в момент начала выполнения программы камера установлена в положение, соответствующее первой точке маршрута.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("TELEMETRY","1.1","PATROL_LEARN","cam<1>,preset<1>,tel_prior<1>,dwell<25>,spee
d<10>,flush_tour<0>");
    Wait(2);
    DoReact("TELEMETRY","1.1","LEFT","speed<6>,tel_prior<1>");
    Wait(2);
    DoReact("TELEMETRY","1.1","STOP","speed<6>,tel_prior<1>");
    Wait(2);
    DoReact("TELEMETRY","1.1","PATROL_LEARN","cam<1>,preset<2>,tel_prior<1>,dwell<25>,spee
d<10>,flush_tour<1>");
}
```

4. Есть 2 камеры с поворотными устройствами. Каждые 15 минут нужно повернуть камеры в пресет №1 (предустановка №1) и сделать скриншот. Имя файла – текущее время.

```
OnTime(W,D,X,Y,H,M, "01")
{
    if(strequal(M,"0"))
    {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотное устройство 1.1
        name1="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
        //Камера 2 Поворотное устройство 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
    }

    if(strequal(M,"15"))
    {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотное устройство 1.1
        name1="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
        //Камера 2 Поворотное устройство 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
    }
```

```
    if(strequal(M,"30"))
    {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотное устройство 1.1
        name1="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
        //Камера 2 Поворотное устройство 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
    }

    if(strequal(M,"45"))
    {
        name=H+"_"+M+"_"+S+".jpg";
        //Камера 1 Поворотное устройство 1.1
        name1="Камера1 "+name;
        DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
        //Камера 2 Поворотное устройство 1.2
        name="Камера2 "+name;
        DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
        DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);
    }
}
```

5. Осуществить патрулирование нескольких зон видимости с помощью пресетов поворотной камеры, с возможностью включения детектора движения на определенных областях этих зон.
Камера 1: 5 зон детектора, 5 предустановок (пресетов). Два этих параметра задаются переменной n.
Макрокоманда №1 – старт алгоритма. Макрокоманда №2 – остановка алгоритма. Flag – внутренняя переменная.
При старте алгоритма камера становится в 1 пресет и ставит на охрану 1 зону детектора. Между этими командами задержка 200 миллисекунд, чтобы камера успела встать в пресет. Далее через 5 секунд 1 зона снимается с охраны и цикл начинается заново, но уже с 2 зоной и 2 пресетом. И так далее пока не переберутся все n зон и пресетов. После алгоритм начинается заново с 1. Алгоритм останавливается, если переменная flag обнуляется (с помощью макрокоманды №2).

```
OnEvent("MACRO","1","RUN")
{
    flag=1;
    n=5;
    [
     for(i=1;flag;i=str(i+1))
     {
         DoReact("TELEMETRY","1.1","GO_PRESET","preset<"+i+">,tel_prior<3>");
         Sleep(200);
         DoReact("CAM_ZONE","1"+i,"ARM");
         Wait(5);
         DoReact("CAM_ZONE","1"+i,"DISARM");
         if(strequal(i,n)) {i=0;}
    }
    ]
}
```

```
OnEvent("MACRO","2","RUN")
{
    flag=0;
}
```

Пример использования событий и реакций объекта **Пульт управления**:

По нажатию клавиши 15 на клавиатуре *AXIS T8312* включить на ней лампочку и поставить камеру 2 на охрану.

```
OnEvent ("TELEMETRY_EXT","1","KEY_PRESSED")
{
    if (strequal(param0, "15")){
        DoReact("TELEMETRY_EXT","1","RELE_ON","rele<15>");
        DoReact("CAM","2","ARM");
    }
}
```

6.3.14 Примеры с Протоколом оператора и Протоколом событий

OPERATORPROTOCOL Протокол оператора
EVENT_VIEWER Протокол событий

6.3.14.1 Форматы

Формат процедуры событий для объекта **Протокол оператора**:

```
OnEvent("OPERATORPROTOCOL","_id_","_событие_")
```

Формат оператора для описания действий с **Протоколом оператора**:

```
DoReact("OPERATORPROTOCOL","_id_","_команда_" ["_параметры_"]);
```

Формат процедуры событий для объекта **Протокол событий**:

```
OnEvent("EVENT_VIEWER","_id_","_событие_")
```

Формат оператора для описания действий с **Протоколом событий**:

```
DoReact("EVENT_VIEWER","_id_","_команда_" ["_параметры_"]);
```

6.3.14.2 Примеры

Примеры использования событий и реакций объекта **Протокол оператора**:

1. По макрокоманде 2 удалять из окна Протокола оператора 1 первую тревогу по Камере 3.

```
OnEvent ("MACRO","2","RUN")
{
    DoReact("OPERATORPROTOCOL","1","DEL_ALARM","objtype<CAM>,objid<3>,options<first>");
```

```
}
```

2. По макрокоманде 2 скрыть в окне Протокола оператора 1 кнопки Тревожная ситуация, Подозрительная ситуация, Ложное срабатывание для события Снята с охраны от Камеры 12.

```
OnEvent ("MACRO","2","RUN")
{
    DoReact("OPERATORPROTOCOL","1","HIDE_BUTTON","button<alarm,suspicious,false>,hide<1>,objtype<CAM>,objaction<DISARM>,objid<12>");
}
```

Пример использования событий и реакций объекта **Протокол событий**:

По макрокоманде 1 для Протокола событий 1 устанавливается основной цвет фона черный, а основной цвет текста – белый.

```
OnEvent ("MACRO","1","RUN")
{
    DoReactStr("EVENT_VIEWER","1","UPDATE_VIEW","bk_color<#000000>, defclr<#FFFFFF>");
}
```

6.3.15 Примеры с Реле и Лучами

GRELE Реле
GRAY Луч

6.3.15.1 Форматы и функции

Формат процедуры событий для **Реле**:

```
OnEvent("GRELE", "_id_","_событие_")
```

Формат оператора для описания действий с **Реле**:

```
DoReact("GRELE","_id_","_команда_");
```

Функция проверки состояния объекта **Реле**:

```
CheckState("GRELE","номер", "состояние")
```

Формат процедуры событий для **Луча**:

```
OnEvent("GRAY","_id_","_событие_")
```

Формат оператора для описания действий с **Лучом**:

```
DoReact("GRAY","_id_","_команда_");
```

Функция проверки состояния объекта **Луч**:

```
CheckState ("GRAY","номер","состояние")
```

6.3.15.2 Примеры

Пример использования событий и реакции объекта **Реле**:

Необходимо при потере связи с реле 1 включить реле 2.

```
OnEvent("GRELE","1","SIGNAL_LOST")
{
    DoReact("GRELE", "2", "ON");
}
```

Примеры использования событий и реакций объекта **Луч**:

1. Необходимо перевести второй луч на второй вход, если потеряна связь с первым лучом.

```
OnEvent("GRAY","1"," SIGNAL_LOST") //потеряна связь с первым лучом
{
    DoReact("GRAY","2","SETUP","chan<2>"); //луч на втором входе
}
```

2. Необходимо разомкнуть второй луч и поставить на запись с откатом первую видеокамеру, в случае, когда первый луч замкнут.

```
OnEvent("GRAY","1"," ON") //первый луч замкнут
{
    DoReact("GRAY","2","SETUP","type<1>"); //разомкнуть второй луч
    DoReact("CAM","1","REC","rollback<1>"); //запись с откатом с первой видеокамеры
}
```

6.3.16 Примеры с Сервером и менеджером инцидентов

INC_MANAGER Менеджер инцидентов
INC_SERVER Сервер инцидентов

Формат процедуры событий для объекта **Менеджер инцидентов**:

```
OnEvent("INC_MANAGER","_id_","_событие_")
```

Формат процедуры событий для объекта **Сервер инцидентов**:

```
OnEvent("INC_SERVER","_id_","_событие_")
```

Формат оператора для описания действий с объектом **Сервер инцидентов**:

```
DoReact("INC_SERVER","_id_","_команда_" [,"_параметры_"]);
```

6.3.17 Примеры с Сервисами сообщений и оповещений

MMS Сервис почтовых сообщений
MAIL_MESSAGE Почтовое сообщение
VMS Сервис голосовых сообщений
VNS Сервис голосового оповещения
SMS Сервис коротких сообщений
TELEGRAM Telegram бот

6.3.17.1 Форматы

Формат процедуры событий для **Сервиса почтовых сообщений**:

```
OnEvent("MMS","_id_","_событие_")
```

Формат оператора для описания действий с **Сервисом почтовых сообщений**:

```
DoReact("MMS","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для **Почтового сообщения:**

```
OnEvent("MAIL_MESSAGE","_id_","_событие_")
```

Формат оператора для описания действий с **Почтовым сообщением**:

```
DoReact("MAIL_MESSAGE","_id_","_команда_" [,"_параметры_"]);
```

Формат оператора для описания действий с **Сервисом голосовых сообщений**:

```
DoReact("VMS","_id_","_команда_" [,"_параметры_"]);
```

Формат оператора для описания действий с **Сервисом голосового оповещения**:

```
DoReact("VNS","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Сервис коротких сообщений**:

```
OnEvent("SMS","_id_","_событие_")
```

Формат оператора для описания действий с **Сервисом коротких сообщений**:

```
DoReact("SMS","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для **Telegram бота**:

```
OnEvent("TELEGRAM","_id_","_событие_")
```

Формат оператора для описания действий с **Telegram ботом**:

```
DoReact("TELEGRAM","_id_","_команда_" ["_параметры_"]);
```

6.3.17.2 Примеры

Пример использования реакций объекта **Сервис почтовых сообщений:**

Необходимо установить номер порта почтовой службы равным 25 при выполнении макрокоманды 1.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("MMS", "1", "SETUP", "port<25>");
}
```

Пример использования реакций объекта **Почтовое сообщение**:

Необходимо отправить сообщение при срабатывании датчика движения вместе с изображением с видеокамеры при переходе видеокамеры в состояние тревоги.

```
OnInit(){
    i=0; //счетчик, используется для того чтобы избежать перезаписывания картинок с одной
камеры
}

OnEvent("CAM",N,"REC") //видеокамера в состоянии тревоги

{
    filename = "c:\" + N + "_msg_"+str(i)+".jpg";
    i=i+1;
    DoReact("MONITOR","1","EXPORT_FRAME","cam<"+ N + ">,file<" + filename+ ">");
    DoReact("MAIL_MESSAGE", "1", "SETUP", "body<сработала камера"+ N + ">, subject<тревога по
камере>, from<server@itv.ru>, to<client@itv.ru>,attachments<" + filename + ">");

    DoReact("MAIL_MESSAGE","1","SEND");

}
```

Пример использования реакций объекта **Сервис голосовых сообщений**:

Необходимо при выполнении макрокоманды 1 послать сообщение, если модем подключен к порту COM2, тип набора – импульсный, не дожидаться тонального сигнала.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("VMS","1","SEND","modem<2>,pulse<1>,waitfordialtone<0>");
}
```

Примеры использования событий и реакций объекта **Сервис голосового оповещения**:

1. Необходимо проигрывать звуковой файл при остановке записи видеокамеры:

```
OnEvent("CAM",N,"REC_STOP")
{
    DoReact("VNS","1","PLAY","file<C:\Program Files (x86)\Intellect\Wav\cam_alarm_"+N+".wa
v>");
```

```
}
```

2. Необходимо завершать проигрывание файла при начале записи видеокамеры:

```
OnEvent("CAM",N,"REC")
{
    DoReact("VNS","1","STOP");
}
```

3. Необходимо, чтобы при наступлении заранее заданной временной зоны менялось значение регулятора громкости на меньшее, а затем по её окончании, ставилось значение равному среднему.

```
OnEvent("TIME_ZONE","1","ACTIVATE")
{
    DoReact("VNS","1","SETUP","level<2>");
}
OnEvent("TIME_ZONE","1","DEACTIVATE")
{
    DoReact("VNS","1","SETUP","level<8>");
}
```

Примеры использования событий и реакций объекта **Сервис коротких сообщений**:

1. Необходимо послать короткое сообщение на номер «89179190909» при тревоге на первой видеокамере.

```
OnEvent("CAM","1","MD_START")
{
    DoReact("SMS","1","SETUP","phone<+79179190909>,message<камера 1, тревога>");
}
```

2. Необходимо установить устройство для передачи коротких сообщений и послать сообщение по номеру «89179190909» при тревоге на первом луче.

```
OnEvent("GRAY","1","CONFIRM") //принять тревогу от луча 1
{
    DoReact("SMS","1","SETUP","device<>",); //установить устройство для передачи коротких
сообщений
    DoReact("SMS","1","SETUP","phone<+79179190909>,message<луч  1, тревога>"); //послать
сообщение о тревоге на луче 1 по номеру телефона
}
```

3. При получении SMS через Сервис почтовых сообщений 2 проиграть звуковой файл c:\Windows\Media\Tada.wav.

```
OnEvent("SMS","2","RECEIVE")
{
    DoReact("PLAYER","3","PLAY_WAV","file<c:\Windows\Media\Tada.wav>");
}
```

Примеры вызова команды для отправки сообщения в **Telegram** по макрокоманде:

```
OnEvent("MACRO","3","RUN") //запуск макрокоманды 3
{
```

```
  //Отправка с использованием chat_id и bot_id из настроек объекта:
  DoReact("TELEGRAM",1,"SEND","text<Hello world>");

  //Явное задание chat_id и bot_id при отправке:
  DoReact("TELEGRAM",1,"SEND","text<Hello
world>,chat_id<828752651>,bot_id<809045046:AAGtKxtDWu5teRGKW_Li8wFBQuJ-l4A9h38>");

  //Отправка файла с указанием идентификаторов чата и бота:
  DoReact("TELEGRAM",1,"SENDPHOTO","caption<Hello
world>,chat_id<828752651>,bot_id<809045046:AAGtKxtDWu5teRGKW_Li8wFBQuJ-l4A9h38>,photo<G:\
\1.jpg>");

  //Отправка геолокации с указанием идентификаторов чата и бота:
  DoReact("TELEGRAM",1,"SEND","text<Hello
world>,chat_id<828752651>,bot_id<809045046:AAGtKxtDWu5teRGKW_Li8wFBQuJ-
l4A9h38>",longtitude<37.3428359>,latitude<55.6841654>,address<ITV>);
}
```

6.3.18 Примеры со Службой перезагрузки системы и Сервисом отказоустойчивости

SSS_WATCHDOG Служба перезагрузки системы
FAILOVER Сервис отказоустойчивости

6.3.18.1 Форматы

Формат процедуры событий для объекта **Служба перезагрузки системы**:

```
OnEvent("SSS_WATCHDOG","_id_","_событие_")
```

Формат оператора для описания действий со **Службой перезагрузки системы**:

```
DoReact("SSS_WATCHDOG","_id_","_команда_" [,"_параметры_"]);
```

Формат процедуры событий для объекта **Сервис отказоустойчивости**:

```
OnEvent("FAILOVER","_id_","_событие_")
```

Формат оператора для описания действий с **Сервисом отказоустойчивости**:

```
DoReact("FAILOVER","_id_","_команда_" [,"_параметры_"]);
```

6.3.18.2 Примеры

Пример использования событий и реакций объекта **Служба перезагрузки системы**:

При перезагрузке модуля активировать третью камеру на монитор 5.

```
OnEvent("SSS_WATCHDOG","1"," RESTART_PROCESS")
{
    DoReact("MONITOR", "5", " ACTIVATE_CAM", "CAM<3>")
```

```
}
```

6.3.19 Примеры с титрами

TITLEVIEWER Поиск по титрам
CAM_TITLE Титрователь

6.3.19.1 Форматы

Формат процедуры событий для объекта **Поиск по титрам**:

```
OnEvent("TITLEVIEWER","_id_","_событие_")
```

Формат оператора для описания действий с **Титрователем**:

```
DoReact("CAM_TITLE","_id_","_команда_");
```

6.3.19.2 Примеры

Пример для **Поиска по титрам**:

При двойном клике по строке результата поиска в окне Поиск по титрам отображать на мониторе 4 видеоархив, соответствующий данному результату.

```
OnEvent("TITLEVIEWER","1","GO_VIDEO")
{
    DoReact("MONITOR","4","ARCH_FRAME_TIME","cam<"+cam+">,date<"+date+">,time<"+time+">");
    DoReact("MONITOR","4","KEY_PRESSED","key<PLAY>");
}
```

Пример для **Титрователя**:

Запускать обновление базы данных титров по макрокоманде 1.

```
OnEvent("MACRO","1","RUN")
{
    DoReact("CAM_TITLE", "2", "REINDEX");
}
```

6.4 Приложение 1. Приоритеты команд начала и остановки записи

Команды остановки и начала записи в ПК *Интеллект* могут иметь разный приоритет. Приоритет команд начала и остановки записи задается параметром priority<> реакций REC и REC_STOP соответственно. В случае, если производится попытка остановить запись командой с меньшим приоритетом, чем у команды, инициировавшей запись, команда на остановку записи будет проигнорирована.

При начале или остановке записи вручную, макрокомандой или по срабатыванию детектора приоритет не указывается явно. В таблице описано поведение ПК *Интеллект* при использовании разных способов начала и остановки записи.

Способ начала/остановки записи 1	Способ начала/остановки записи 2	Поведение

Начало/остановка записи инициированы оператором с помощью контекстного меню камеры (запись/остановить запись) или макрокомандой	Начало записи по реакции CAM|1|REC|, остановка по реакции CAM|1|REC_STOP|	Команды начала и остановки записи первого и второго способа равноценны*
Начало/остановка записи инициированы оператором с помощью контекстного меню камеры (запись/остановить запись) или макрокомандой	Начало записи по реакции CAM|1|REC|priority<0>, остановка по реакции CAM|1|REC_STOP|priority<0>	Остановка записи первым способом останавливает запись, начатую вторым способом
Начало/остановка записи инициированы оператором с помощью контекстного меню камеры (запись/остановить запись) или макрокомандой	Начало записи по реакции CAM|1|REC|priority<1> , остановка по реакции CAM|1|REC_STOP|priority<1>	Команда остановки записи первым способом останавливает запись, начатую вторым способом
Начало/остановка записи инициированы оператором с помощью контекстного меню камеры (запись/остановить запись) или макрокомандой	Начало записи по реакции CAM|1|REC|priority<2>, остановка по реакции CAM|1|REC_STOP|priority<2>	Команды начала и остановки записи первого и второго способа равноценны*
Начало/остановка записи инициированы оператором с помощью контекстного меню камеры (запись/остановить запись) или макрокомандой	Запись/остановка записи инициирована детектором (например, базовым детектором движения)	Команда остановки записи первым способом останавливает запись, начатую вторым способом
Начало записи по реакции CAM|1|REC|priority<0>, остановка по реакции CAM|1|REC_STOP|priority<0>	Запись/остановка записи инициирована детектором (например, базовым детектором движения)	Команда остановки записи вторым способом останавливает запись, начатую первым способом
Начало записи по реакции CAM|1|REC|priority<1>, остановка по реакции CAM|1|REC_STOP|priority<1>	Запись/остановка записи инициирована детектором (например, базовым детектором движения)	Возможны следующие варианты: 1. Если камера на охране и осуществляется запись по команде CAM|1|REC|priority<1>, при этом происходит начало тревоги по камере, то после окончания тревоги по камере запись продолжается. После команды CAM|1|REC_STOP|priority<1> запись оканчивается. 2. Если камера на охране, инициирована тревога по камере, а затем послана команда CAM|1|REC|priority<1>, после чего тревога по камере закончилась, то запись продолжается. После команды CAM|1|REC_STOP|priority<1> запись оканчивается. 3. Если камера на охране, инициирована тревога по камере, а затем послана команда CAM|1|REC_STOP|priority<1> то запись продолжается, а по окончании тревоги по камере запись оканчивается. 4. Если камера на охране и послана команда CAM|1|REC|priority<1>, начнется запись по камере. Если после этого будет инициирована тревога по камере и будет послана

		команда CAM|1|REC_STOP|priority<1>, то запись продолжится.
Начало записи по реакции CAM|1|REC|priority<2>, остановка по реакции CAM|1|REC_STOP|priority<2>	Запись/остановка записи инициирована детектором (например, базовым детектором движения)	Команда остановки записи первого способа останавливает запись, начатую вторым способом

*Равноценность способов означает, что остановка записи способом 1 возможна, если запись инициирована способом 2, и наоборот, остановка записи способом 2 возможна, если запись инициирована способом 1

6.5 Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM

Значения параметров **param_id** и **param_value**, необходимых для использования реакции SET_IPINT_PARAM, могут быть индивидуальны как для каждой из интегрированных IP-камер, так и для их прошивок.

Определение значений **param_id** и **param_value** осуществляется следующим образом:

1. Открыть директорию с установленным DriverPack, по умолчанию **C:\Program Files\Common Files\ITV\Ipint.DriverPack\3.0.0**
2. Открыть с помощью любого текстового редактора содержащийся в данном каталоге файл с именем **Ipint.<Название драйвера камеры>.rep**, например Ipint.SonyIpela.rep

   Примечание.

   В большинстве случаев имя драйвера совпадает с названием производителя IP-устройства. Уточнить имя драйвера для требуемого производителя можно при обращении в техническую поддержку компании ITV.

3. Найти в файле название требуемой модели, например SNC-DH120T.

```
    <model>
        <brand>Sony</brand>
        <name>SNC-DH120T</name>
        <firmware>1.12.03</firmware>
        <firmware>1.74.01</firmware>
        <firmware>1.75.00</firmware>
    </model>
    <credentialsRef id="creds"/>
    <videoSourceRef id="video_source_dh160">
        <videoStreamingRef id="vs-5generation-megapixel-tvStandard" default="true"/>
        <videoStreamingRef id="vs-5generation-secondary-ch120"/>
        <detectorRef id="sony-detector-area-1280x1024" maxCount="1"/>
        <detectorRef id="sony-detector-tamper" maxCount="1"/>
    </videoSourceRef>
    <telemetryRef id="telemetry_5g"/>
    <ioDeviceRef id="iodev-sony-1ray-1relay"/>
</device>
```

4. В пределах того же тэга <device>, что и тэг <model>, содержащий описание требуемой модели, присутствует тэг **<videoSourceRef>**. Необходимо найти в файле еще одно вхождение значения **id** данного параметра (в

данном примере это значение video_source_dh160) в тэге **videoSource**.

```
<videoSource id="video source dh160">
    <property id="brightness" xsi:type="PropertyIntRangeType">
        <value>
            <min>0</min>
            <max>10</max>
            <default>5</default>
        </value>
    </property>
    <property id="sharpness" xsi:type="PropertyIntRangeType">
        <value>
            <min>0</min>
            <max>6</max>
            <default>3</default>
        </value>
    </property>
    <property id="saturation" xsi:type="PropertyIntRangeType">
        <value>
            <min>0</min>
            <max>6</max>
            <default>3</default>
        </value>
    </property>
    <property id="contrast" xsi:type="PropertyIntRangeType">
        <value>
            <min>0</min>
            <max>6</max>
            <default>3</default>
        </value>
    </property>
    <property id="monochrome" xsi:type="PropertyBoolType" default="false"/>
    <property id="daynight" xsi:type="PropertyStringEnumType">
        <value default="true">auto</value>
        <value name="night">on</value>
        <value name="day">off</value>
        <value name="timer">timer</value>
        <value name="sensor">sensor</value>
    </property>
    <property id="dayNightAutoThreshold" xsi:type="PropertyStringEnumType">
        <value name="high" default="true">high</value>
        <value name="low">low</value>
    </property>
```

5. В тэгах **<property>** описаны параметры IP-устройства и их возможные значения. Способ описания возможных значений зависит от их типа.

В приведенном примере можно использовать, например, параметр **param_id**="daynight" для переключения режима камеры **День/Ночь**. В таком случае возможные значения параметра **param_value**: auto, on, off, timer или sensor.

Пример

Пример использования реакции SET_IPINT_PARAM:

1. Для объекта **Камера**:
   DoReact("CAM", "1","SET_IPINT_PARAM","param_id<daynight>,param_value<on>");
2. Для объекта **Устройство видеоввода**:
   DoReact("GRABBER", "1","SET_IPINT_PARAM","param_id<daynight>,param_value<on>,cam_id<1>");

Результатом выполнения обеих реакций будет установка значения параметра "daynight" для Камеры 1 равным "on".

Для работы реакции SET_IPINT_PARAM необходимо, чтобы в ПК *Интеллект* был активирован многопоточный режим (см. Руководство администратора, раздел Настройка многопоточного видеосигнала). При этом следует учитывать, что если для камеры интегрирован только один видеопоток, в многопоточном режиме не будет отображаться видеоизображение.

Узнать количество интегрированных потоков для камеры можно в списке IP-оборудования, интегрированного в ПК *Интеллект*, который находится на странице Documentation Drivers Pack.

Если данный способ неприменим по каким-либо причинам, количество интегрированных видеопотоков можно узнать следующим образом:

1. Повторить шаги 1-3 предыдущего алгоритма.
2. В пределах того же тэга <device>, в котором описана требуемая модель, в тэгах **<videoStreamingRef>** описаны интегрированные видеопотоки. Их должно быть больше одного.

```
    <model>
        <brand>Sony</brand>
        <name>SNC-DH120T</name>
        <firmware>1.12.03</firmware>
        <firmware>1.74.01</firmware>
        <firmware>1.75.00</firmware>
    </model>
    <credentialsRef id="creds"/>
    <videoSourceRef id="video_source_dh160">
        <videoStreamingRef id="vs-5generation-megapixel-tvStandard" default="true"/>
        <videoStreamingRef id="vs-5generation-secondary-ch120"/>
        <detectorRef id="sony-detector-area-1280x1024" maxCount="1"/>
        <detectorRef id="sony-detector-tamper" maxCount="1"/>
    </videoSourceRef>
    <telemetryRef id="telemetry_5g"/>
    <ioDeviceRef id="iodev-sony-1ray-1relay"/>
</device>
```

7 Описание событий и реакций объектов системы

В данной главе указаны все реакции для основных объектов системы.

Примечание.

События для объектов системы можно просмотреть одним из следующих способов:

1. Просмотр содержимого файла intellect.ddi посредством утилиты «ddi.exe» (см. Получение списка системных названий объектов, реакций и событий ПК Интеллект).
2. Просмотр событий для выбранного объекта системы посредством панели настроек системного объекта **Макрокоманда** (см. Создание и использование макрокоманд).

.

7.1 GRABBER Устройство видеоввода

Объект **GRABBER** соответствует системному объекту **Устройство видеоввода**.

От объекта **GRABBER** поступают события, представленные в таблице. Запуск процедуры происходит при возникновении соответствующего события.

Описание событий от объекта **GRABBER**:

События	**Описание события**
+12V	Ошибка напряжения +12V
+3.3V	Ошибка напряжения +3.3V
+5V	Ошибка напряжения +5V
-12	Ошибка напряжения -12V
-5V	Ошибка напряжения -5V
CPU_FAN	Количество оборотов вентилятора
CPU_TEMP	Температура процессора
SYS_TEMP	Температура чипсета MB
UPS_COMMLOST	Потеря связи
UPS_FATAL_ERROR	Ошибка подключения
UPS_LOWBATT	Села батарея
UPS_ONBATT	Переход на питание от батареи
UPS_ONLINE	Восстановление питания от сети
UPS_REPLACEBATT	Требуется замена батареи

События	**Описание события**
UPS_SHUTTING	Выключение
VCORE	Напряжение ядра процессора
AUDIO_SIG_LOST	Потеря звука
CONNECT_FAIL	Ошибка подключения
CONNECT_OK	Подключено
NETWORK_FAILURE	Соединение потеряно
STATE_CONNECTED	Соединение восстановлено

Список команд и параметров для объекта **GRABBER** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – устанавливает параметры устройства видеоввода	chan<>	Номер PCI слота (0,1,2,…,32)
	mode<>	Скорость граббера/оцифровки (0 – максимальная, 1 – средняя, 2 – минимальная)
	resolution<>	Разрешение: 0 – стандартное, четверть кадра (384x288); 1 – высокое, полукадр (768x288); 2 – максимальное, кадр (768x576)
	format<>	Формат видеосигнала (PAL, NTSC)
	drives<>	Диски для записи видеоархива (DRIVE1:\, DRIVE2:\ … DRIVEN:\)
	cams<>	Количество подключенных видеокамер
	auth<>	Данные авторизации
	ip<>	IP-адрес сетевой платы видеоввода
	name<>	Имя объекта
	flags <>	Флаги
	ip_port<>	IP-порт
	password<>	Пароль
	type<>	Тип оцифровки
	username<>	Логин
	watchdog<>	Включение WatchDog (0 – выключен, 1 – включен)

Команда – описание команды	Параметры	Описание параметров
"SET_DRIVES" – устанавливает диски для записи видеоархива	drives<>	Диски для записи видеоархива
"MUX1_OFF" – отключить вывод видео через аналоговый выход 1	-	-
"MUX2_OFF" – отключить вывод видео через аналоговый выход 2	-	-
"MUX3_OFF" – отключить вывод видео через аналоговый выход 3	-	-
"SET_IPINT_PARAM" – Установить (изменить) параметры IP-устройства. Реакция позволяет менять настройки IP-устройства, не заходя в его web-интерфейс. *Примечание. Для работы реакции необходимо включить режим многопоточного видеосигнала (см.* Руководство администратора*, раздел* Настройка многопоточного видеосигнала*), а также* Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM	param_id<>	Название параметра. Набор параметров для каждой камеры индивидуален – см. Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM
	param_value<>	Значение параметра. Набор параметров для каждой камеры индивидуален – см. Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM
	cam_id<>	Идентификатор камеры в ПК *Интеллект*
	vstream_id<>	Номер видеопотока (не обязательный параметр). Имеет вид "Номер камеры"."Номер потока", например 1.1, 1.2.
"START" – начать проигрывание видеоролика в виртуальном устройстве видеоввода	-	-
"STOP" – остановить проигрывание видеоролика в виртуальном устройстве видеоввода	-	-
"ENABLE" – включить (снять флажок **Отключить** на панели настройки объекта)	recursive<>	Возможные значения параметра: 0 – включить только **Устройство видеоввода** 1 – включить **Устройство видеоввода** и все объекты **Камера**, созданные на базе него
"DISABLE" – отключить (установить флажок **Отключить** на панели настройки объекта)	recursive<>	Возможные значения параметра: 0 – отключить только **Устройство видеоввода** 1 – отключить **Устройство видеоввода** и все объекты **Камера**, созданные на базе него

Свойства объекта **GRABBER** показаны в таблице:

Свойства объекта GRABBER	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Номер устройства видеоввода

7.2 CAM Камера

Объект **CAM** соответствует системному объекту **Камера**.

От объекта **CAM** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

Описание событий от объекта **CAM**:

События	**Описание событий**	**Комментарий**
ARM	Камера поставлена на охрану	Если постановка на охрану выполнена Оператором с **Карты** или из **Монитора видеонаблюдения**, в параметре user_id<> содержится идентификатор пользователя, выполнившего это действие
ATTACH	Подключение	
BLINDING	Камера залеплена	
DETACH	Обрыв	Событие генерируется при потере входного сигнала с камеры на плате видеоввода
DISARM	Камера снята с охраны	Если снятие с охраны выполнено Оператором с **Карты** или из **Монитора видеонаблюдения**, в параметре user_id<> содержится идентификатор пользователя, выполнившего это действие
FILE_REC_ERROR	Ошибка записи на диск	Событие генерируется, когда происходит ошибка записи видеоархива на диск
MD_START	Тревога	
MD_STOP	Конец тревоги	
PRINT	Печать кадра	
REC	Запись на диск	Если запись инициирована Оператором с **Карты** или из **Монитора видеонаблюдения**, в параметре user_id<> содержится идентификатор пользователя, выполнившего это действие
REC_STOP	Остановка записи на диск	Если запись остановил Оператор с **Карты** или из **Монитора видеонаблюдения**, в параметре user_id<> содержится идентификатор пользователя, выполнившего это действие
UNBLINDING	Камера открыта	
RECORDER_ON	Запись включена	
RECORDER_OFF	Запись выключена	
DISC_MOUNT	Диск подмонтирован	
DISC_UNMOUNT	Диск отмонтирован	
FINISHED_AVI_EXPORT	Экспорт видео завершен	В случае, если попытка экспорта видео завершилась неудачей, событие имеет ненулевой параметр error_result. В параметре param<0> содержится дополнительная информация, отображаемая в соответствующем столбце в **Протоколе событий** в формате "ИмяКомпьютера;ПериодЭкспорта;ИмяПользователя;ИдентификаторПользователя" – например, param0<LOCALHOST;04-06-18 16:50:55_04-06-18 16:55:55;;1>

События	**Описание событий**	**Комментарий**
MD_LIMIT	Превышено количество объектов в кадре	Событие генерируется при превышении количества обнаруженных трекером объектов в кадре (см. Создание и настройка объекта Трекер). Событие генерируется при каждом изменении количества объектов (в большую или меньшую сторону), пока оно не будет меньше пороговой. В параметре object_count<> передается число объектов в кадре, которое обнаружено трекером, превышает заданный лимит и отличается от предыдущего значения. См. также описание события NEW_OBJECT ниже
NEW_OBJECT	Трекер обнаружил новый объект в кадре	Помимо прочих, содержит следующие параметры: • total<> – общее количество объектов в кадре на момент возникновения события • new_id<> – идентификатор обнаруженного объекта
ARCH_DELETE	Удаление записи	Удаление записи архива по камере из **Монитора видеонаблюдения**
OPEN_FILE	Открытие файла	Открытие файла для воспроизведения виртуальным устройством видеоввода. Свидетельствует о начале воспроизведения следующего файла в выбранной папке. Помимо прочих, содержит следующие параметры: • name<> – название проигрываемого файла • tss<> – время в формате UTC в миллисекундах с 01.01.1970
CLOSE_FILE	Закрытие файла	Завершение воспроизведения файла виртуальным устройством видеоввода. Свидетельствует об окончании воспроизведения файла в выбранной папке. Помимо прочих, содержит следующие параметры: • name<> – название файла, проигрывание которого завершилось • tss<> – время в формате UTC в миллисекундах с 01.01.1970
ARCH_PROTECTED	Файл защищен от перезаписи	Событие отображается, когда пользователь защищает файл от перезаписи. В параметре param<0> содержится дополнительная информация об имени компьютера, фрагменте, имени и id пользователя, отображаемая в столбце **Доп. инфо** в **Протоколе событий** в формате "ИмяКомпьютера;ЗащищенныйПериод;ИмяПользователя;ИдентификаторПользователя" – например, param0<LOCALHOST;04-06-18 16:50:55.612_04-06-18 16:55:55.612;User 1;1>
ARCH_UNPROTECTED	Снята защита файла от перезаписи	Событие отображается, когда пользователь снимает защиту файла от перезаписи. В параметре param<0> содержится дополнительная информация об имени компьютера, фрагменте, имени и id пользователя, отображаемая в столбце **Доп. инфо** в **Протоколе событий** в формате "ИмяКомпьютера;ЗащищенныйПериод;ИмяПользователя;ИдентификаторПользователя" – например, param0<LOCALHOST;04-06-18 16:50:55.612_04-06-18 16:55:55.612;User 1;1>
FRAME_SKIPPED	Пропуски кадров	Событие FRAME_SKIPPED поступает в ПК *Интеллект*, если есть пропуски кадров при записи в архив. По умолчанию событие генерируется, если за период проверки было более 50 пропусков кадров. Когда пропуски кадров прекращаются, генерируется событие FRAME_SKIPPED_STOP. В описании этого события содержится информация о количестве пропущенных кадров и промежутке времени, в который были пропуски. Оба события по умолчанию генерируются не чаще чем раз в 30 секунд. События регулируются следующими ключами реестра (см. Справочник ключей реестра): • событие FRAME_SKIPPED можно отключить с помощью ключа реестра FileSystem.NotifyCoreFrameSkipped; • время задержки между сменой состояний в секундах задаётся в ключе FileSystem.RecordingStateChangeDelay;

События	Описание событий	Комментарий
FRAME_SKIPPED_STOP	Конец пропуска кадров	• количество пропущенных кадров за период, при котором будет генерироваться FRAME_SKIPPED, задаётся в ключе FileSystem.MaxSkippedFramesByPeriod.
ARCH_BOOKMARKED	Создана закладка	Событие отображается, когда пользователь добавляет закладку. В параметре param<0> (столбце **Доп. инфо** в **Протоколе событий**) содержится комментарий к закладке
ARCH_UNBOOKMARKED	Удалена закладка	Событие отображается, когда пользователь удаляет закладку
TEMPERATURE_ALARM	Порог температуры	В параметре param0<> содержится значение температуры, полученное от тепловизора
IGNORE_KEEP_NO_LESS	Игнорирование "Хранить не менее"	Событие генерируется при удалении из архива видеозаписей, которые должны были бы храниться в течение более длительного периода времени, задаваемого параметром **Хранить не менее**. См. также Настройка глубины архива по видеокамере
AVAILABILITY	Наличие ключевой позиции в файле лицензии и количество объектов	Событие приходит в ответ на команду GET_AVAILABILITY и содержит информацию о количестве добавленных в ключ лицензии объектов для заданной позиции. Параметры: • qty_localpriority<> – количество разрешенных объектов заданного типа на этом компьютере (локальная часть ключа) • qty<> – общее количество разрешенных объектов заданного типа в лицензии (общая часть ключа) • pos<> – номер позиции в ключе лицензии Если позиция в ключе отсутствует, то событие будет со значением ноль
WRITING_FAILED	Ошибка записи	
FILESYSTEM_FAILED	Ошибка файловой системы	Содержит обязательный параметр error_msg<> **–** текст ошибки; также может содержать параметры cam<> **–** идентификатор камеры и error_code<> **–** код ошибки. Пример: CAM|-1|FILESYSTEM_FAILED|error_msg<Failed to delete index file: D:\\VIDEO\\INDEX\\14061608.idx, error code: 5.>,error_code<5>

Список команд и параметров для объекта **CAM** представлен в таблице.

Команда – описание команды	Параметры	Описание параметров
"SETUP" – устанавливает (изменяет) параметры камеры	rec_priority<>	Приоритет записи (от 0 до 3, 0 – обычный, 3 – все ресурсы)
	compression<>	Степень компрессии (0 – компрессия отсутствует, 1- макс. качество, …, 5 – мин. качество)
	sat_u<>	Насыщенность цвета (0 – мин, 10 – макс)
	proc_time<>	Период дозаписи (0-30 сек)
	manual<>	Управление настройкой яркости и контрастности (0 – ручное; 1 – автоматическое; 2 – автоматическое, но около значений, выставленных вручную)
	contrast<>	Контрастность (0 – мин, 10 – макс)

Команда – описание команды	Параметры	Описание параметров
	md_size<>	Размер объектов детектора движения (1-16)
	md_mode<>	Режим записи пауз (1 – включено, 0 – выключено)
	audio_type<>	Тип звукового сопровождения
	pre_rec_time<>	Время предзаписи (0-20 сек)
	bright<>	Яркость (0 – мин, 10 – макс)
	audio_id<>	Номер микрофона (пустой параметр, если нет микрофона)
	rec_time<>	Скорость записи (1-30 кадров/сек, 0 – не используется)
	alarm_rec<>	Запись тревог (1 – включено, 0 – выключено)
	hot_rec_time<>	Время горячей записи (0-30 сек)
	hot_rec_period<>	Период горячей записи (0-20 сек)
	mux<>	Номер канала (0 – 1 канал, 15 – 16 канал)
	color<>	Цвет (0 – черно-белый, 1 – цветной)
	activity<>	-
	arch_days<>	Количество дней архива
	blinding<>	Камера залеплена
	config_id<>	-
	decoder<>	-
	flags<>	Флаги
	fps<>	Скорость записи (0 – не используется, 1 – 30 кадров/сек)
	ifreq<>	Частота опорных кадров в последовательности (1 – каждый кадр опорный, 2 – 100 кадр)
	mask 0, mask1, mask2, mask3, mask4	Маска детектора
	md_contrast<>	Чувствительность детектора движения (0-15)
	motion<>	Оценка движения компрессора (5-255)

Команда – описание команды	Параметры	Описание параметров
	name<>	Имя объекта
	password_crc<>	Пароль на видеоархив
	priority<>	Приоритет источника постановки на запись (0 – автоматическое, 1 – ручное)
	resolution<>	Разрешение (0 – стандартное CIF, 1 – высокое 2CIF, 2 – максимальное 4CIF)
	type<>	Тип объекта
	yuv<>	Цветовая схема кодирования видеосигнала (0 – YUV4:2:0, 1 – YUV4:2:2)
"DELETE" – отключает камеру	-	-
"START_VIDEO" – включает видеопоток для текущей камеры	slave_id<>	Имя компьютера, к которому подключена камера
	comress<>	Степень компрессии
	register_only<>	-
"STOP_VIDEO" – выключает видеопоток для текущей камеры	slave_id<>	Имя компьютера, к которому подключена камера
"REQUEST_MASK"	mask<>	Маска
"MUX1", "MUX2", "MUX3" – вывести изображение камеры на 1, 2, 3 аналоговые выходы	-	-
"ACTIVATE" – вывести камеру на монитор	monitor<>	Номер монитора
"ARM" – поставить камеру на охрану	-	-
"DISARM" – снять камеру с охраны	-	-
"REC" – начать запись камеры	time<>	Время записи в секундах, если равно нулю, то записывается 1 кадр
	rollback<>	Если равно 1, то запись производится с откатом
	priority<>	Задает приоритет команды начала записи. Подробнее см. Приложение 1. Приоритеты команд начала и остановки записи
	stream_id<>	Задает номер потока для записи. Номер потока имеет вид "n.m", где n – id камеры, m – номер потока. *Примечание. Если указанный поток не используется ни по какому другому назначению, кроме записи по команде (и по выбору на клиенте), необходимо убедиться, что для него не установлен флажок* ***Блокировать отключение неиспользуемых потоков*** *– см.* Панель настройки объекта Камера

Команда – описание команды	**Параметры**	**Описание параметров**
"REC_STOP" – остановить запись камеры	priority<>	Задает приоритет команды остановки записи. Подробнее см. Приложение 1. Приоритеты команд начала и остановки записи
	user_id<>	Если остановка записи была выполнена пользователем из **Монитора видеонаблюдения**, то в данном параметре передается идентификатор пользователя. В противном случае данный параметр отсутствует
	from_macro<>	Если остановка записи была выполнена по макрокоманде, то в данном параметре передается идентификатор макрокоманды. В противном случае данный параметр отсутствует
"SET_MASK" – установить маску	mask<>	Маска
"ADD_SUBTITLES" – добавить титры	command<>	Текст накладываемых титров
	title_id<>	Идентификатор объекта **Титрователь**, используемого для наложения титров
	page<>	Обязательный параметр при записи титров в базу данных титров для обеспечения в дальнейшем возможности поиска по титрам. Возможные значения: BEGIN (начало записи в базе), END (конец записи в базе)
"SIP_CONNECT" – Sip подключен	-	-
"SIP_DISCONNECT" – Sip отключен	-	-
"SET_IPINT_PARAM" – Установить (изменить) параметры IP-устройства. Реакция позволяет менять настройки IP-устройства не заходя в его web-интерфейс. *Примечание. Для работы реакции необходимо включить режим многопоточного видеосигнала – см.* Настройка многопоточного видеосигнала*, а также* Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM	param_id<>	Название параметра. Набор параметров для каждой камеры индивидуален – см. Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM
	param_value<>	Значение параметра. Набор параметров для каждой камеры индивидуален – см. Приложение 2. Определение значений param_id и param_value для реакции SET_IPINT_PARAM
	vstream_id<>	Номер видеопотока (необязательный параметр). Имеет вид "Номер камеры"."Номер потока", например 1.1, 1.2
"GET_FRAME" – получить кадр с камеры, даже если она не отображается на **Мониторе видеонаблюдения**	path<>	Путь для сохранения кадра. Если параметр отсутствует, в системе будет сформировано событие FRAME_SENT с параметром data. Обработка данного события описана в разделе Метод SaveToFile
	stream<>	Необязательный параметр. Задает поток в ПК *Интеллект*, с которого требуется получить кадр. Поток можно указать по номеру или по назначению. Возможные назначения: • stream_archive – поток для записи в архив • stream_alarm – поток для записи в архив по тревогам • stream_client – поток для отображения • stream_analytic – поток для видеоаналитики Номер потока состоит из идентификатора камеры и номера потока, например, 4.3 – третий поток с камеры 4
	time<>	Необязательный параметр. Указывается, если требуется запросить кадр архива. Формат значения параметра: ДД-ММ-ГГГГ ЧЧ:ММ:СС – например, time<19-09-2017 11:35:34>
	gate<>	Необязательный параметр. Задает сетевое имя **Видеошлюза**, из архива которого следует получать кадр

Команда – описание команды	Параметры	Описание параметров
	arch<>	Необязательный параметр. Задает сетевое имя **Долговременного архива**, из которого следует получать кадр
	slave_id<>	Необязательный параметр. Задает сетевое имя Сервера, из архива которого следует получать кадр
	password_crc<>	Необязательный параметр. Задает CRC-последовательность, которая будет записана в результирующий файл, содержащий запрошенный кадр
"ARCH_DEL_RECORD" – удалить записи архива за определенный период	fromTime<>	Обязательный параметр. Время в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС.НННН, где НННН – миллисекунды. Будут удалены записи, начиная с первой, содержащей указанный момент времени, и заканчивая последней, содержащей момент времени toTime. Если не указано время в параметре toTime, будет удалена только одна запись
	toTime<>	Необязательный параметр. Время в формате ГГГГ-ММ-ДДТЧЧ:ММ:СС.НННН, где НННН – миллисекунды. Описание см. выше
"REC_RESTART" – перезапустить запись по камере	-	-
"ARCH_BOOKMARK_RECORD" – создать закладку	time1<>	Дата начала периода архива, включенного в закладку, в формате ДД-ММ-ГГ ЧЧ:ММ:СС.НННН, где НННН – миллисекунды
	time2<>	Дата окончания периода архива, включенного в закладку, в формате ДД-ММ-ГГ ЧЧ:ММ:СС.НННН, где НННН – миллисекунды
	comment<>	Комментарий к закладке
	slave_id<>	Идентификаторы компьютера и **Монитора видеонаблюдения**, с использованием которых будет выполняться создание закладки. Формат параметра: <имя компьютера>.<айди монитора>. Например, slave_id<WS2.1> – здесь WS2 является идентификатором компьютера, а 1 – идентификатором Монитора видеонаблюдения
"CRUISE_START" – автопанорамирование	cruise_id<>	Номер маршрута на камере
	action<>	Выполняемое действие: • CRUISE_START – начать автопанорамирование по указанному маршруту • PATROL_PLAY – начать патрулирование по указанному маршруту
	cam_id<>	Идентификатор камеры
"GET_DEPTH" – получить глубину архива. В ответ на реакцию в системе формируется событие ARCHIVE_DEPTH от объекта SLAVE (см. SLAVE Компьютер). Отсутствие одного или обоих параметров означает запрос глубины по записям для всех возможных значений параметра	drive<>	Диск или сетевой путь, по которому запрашивается глубина архива. Название диска задается формате "<буква диска>:\\", например drive<D:\\> *Примечание. Символ "\" – экранируемый.* Сетевой путь задается в формате UNC
	arch	Указывает, что требуется запросить глубину долговременного архива. Пример. `DoReactStr("CAM","2","GET_DEPTH","drive<D:\\>,cam<2>,arch");`
	gate	Указывает, что требуется запросить глубину архива видеошлюза. Пример.

Команда – описание команды	Параметры	Описание параметров
		`DoReactStr("CAM","1","GET_DEPTH","drive<V:\ \>,gate");`
"CLEAR_SUBTITLES" – удалить все титры с видеоизображения	title_id<>	Идентификатор объекта **Титрователь**
"GET_AVAILABILITY" – проверить наличие ключевой позиции	pos<>	Номер позиции в ключе лицензии

Свойства объекта **CAM** показаны в таблице:

Свойства объекта CAM	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта
TELEMETRY_ID<>	Идентификатор модуля телеметрии (ID поворотника)
REGION_ID<>	Идентификатор региона

Объект **CAM** может находиться в состояниях, описанных в таблице:

Состояние объекта CAM	Описание состояния
"ALARMED"	Камера находится в тревожном состоянии
"DISARM_DETACHED"	Нет сигнала от камеры
"DETACHED"	Нет сигнала от камеры
"ARMED"	Камера поставлена на охрану
"DISARMED"	Камера снята с охраны

7.3 MONITOR Монитор видеонаблюдения

Объект **MONITOR** соответствует системному объекту **Монитор**.

От объекта **MONITOR** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

Описание событий от объекта **MONITOR**:

Событие	Описание события	Комментарий
STARTED_AVI_EXPORT	Начало экспорта видео	Среди прочих, событие содержит следующие параметры: • slave_id<> – оператор, запустивший экспорт • param1<> – номер камеры, по которой осуществляется экспорт, а также дата и время начала периода экспорта. Параметр принимает значение вида "<№записи> Камера <id> (дд-мм-гг чч:мм:сс)", например param1<01 Камера 1 (05-10-17 10:23:21)>

		• time<> – время начала экспорта
FINISHED_AVI_EXPORT	Конец экспорта видео	Среди прочих, событие содержит следующие параметры: • slave_id<> – оператор, запустивший экспорт • param1<> – номер камеры, по которой осуществляется экспорт, а также дата и время окончания периода экспорта. Параметр принимает значение вида "<№записи> Камера <id> (дд-мм-гг чч:мм:сс)", например param1<01 Камера 1 (05-10-17 10:40:21)> • time<> – время окончания экспорта • param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, в формате: "ИмяКомпьютера;ПериодЭкспорта;ИмяПользователя;ИдентификаторПользователя", например, param0<LOCALHOST;04-06-18 16:50:55_04-06-18 16:55:55;;1>
AVI_EXPORT_RESULT	Результат экспорта видео	Событие имеет те же параметры, что и START_AVI_EXPORT, с добавлением параметра error_result<>, принимающего одно из следующих значений: 0 – экспорт успешно выполнен 1 – неизвестно 2 – задача занята 3 – не готово 4 – неверный интервал 5 – ошибка файла
PLAY_START	Начало проигрывания фрагмента архива	
PLAY_STOP	Конец проигрывания фрагмента архива	
INTERFACE_MANIPULATION	Изменение визуализации	param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, содержит идентификатор перемещенной по раскладке камеры
LAYOUT_DEL	Удаление раскладки	param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, содержит имя удаленной раскладки
LAYOUT_ADD	Добавление раскладки	param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, содержит имя добавленной раскладки
LAYOUT_ACTIVATE	Смена активной раскладки	param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, содержит имя активированной раскладки
REPLACE_CAM	Смена положения камеры	param<0> – дополнительная информация, отображаемая в соответствующем столбце в Протоколе событий, в формате: <Имя камеры 1> → <Имя камеры 2>
ACTIVATE_CAM	Активация камеры	auto_switch<> – указывает, было ли включено автоматическое листание во время активации камеры. Можно использовать для отключения автоматического листания при активации окна видеонаблюдения
CAM_EXPAND	Увеличение Окна видеонаблюдения на весь Монитор	Событие генерируется только если установлены следующие ключи реестра (см. Справочник ключей реестра): • MaximizeCameraOnDblClk=1 • MinimizeCameraOnDblClk=1 Параметры события: • param0<> – идентификатор камеры • user_id<> – идентификатор пользователя, выполнившего действие

CAM_COLLAPSE	Уменьшение Окна видеонаблюдения	Событие генерируется только если установлены следующие ключи реестра (см. Справочник ключей реестра): • MaximizeCameraOnDblClk=1 • MinimizeCameraOnDblClk=1 Параметры события: • param0<> – идентификатор камеры • user_id<> – идентификатор пользователя, выполнившего действие

Список команд и параметров для объекта **MONITOR** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"REMOVE" – удаляет камеру с монитора	cam<>	ID камеры в дереве настроек, которую необходимо удалить с монитора
	show<>	Необязательный параметр. Возможные значения: • 0 – не обновлять раскладку в Мониторе после удаления камеры. Может оставаться пустое пространство, не занятое Окнами видеонаблюдения • 1 – обновлять раскладку в Мониторе после удаления камеры, чтобы минимизировать пустое пространство
"REMOVE_ALL" – удаляет все камеры с монитора	-	-
"STOP_VIDEO" – останавливает видеопоток камеры	cam<>	ID камеры в дереве настроек, видеопоток от которой необходимо остановить
"REPLACE" – удаляет все камеры с монитора и вызывает указанную камеру	slave_id<>	Имя компьютера, которому принадлежит монитор, в скрипте можно подставить owner
	cam<>	ID камеры в дереве настроек, которую необходимо вывести на монитор
	name<>	Название камеры, которое будет отображаться в левом нижнем углу
	audio_type<>	-
	audio_id<>	-
	arch_id<>	-
	control<>	0 – только просмотр архива, 1 – также возможно и управление (постановка/снятие с охраны, запись)
"ADD_SHOW" – добавляет камеры на монитор *Примечание. См. также PLACE_CAM_IN_LAYOUT_CELL*	cam<>	ID камеры в дереве настроек, которую необходимо вывести на монитор
	name<>	Имя объекта, которое будет отображаться в левом нижнем углу
	arch_id<>	-
	control<>	0 – только просмотр архива, 1 – также возможно и управление (постановка/снятие с охраны, запись)

Команда – описание команды	Параметры	Описание параметров
	gate_id<>	Идентификатор видеошлюза, через который необходимо получать видео для отображения. Соответствующая камера должна быть добавлена и настроена в данном видеошлюзе – см. Выбор и настройка видеокамер для модуля Видеошлюз
	slave_id<>	Идентификатор компьютера, к которому применяется команда
"ACTIVATE_CAM" – делает активной камеру	cam<>	ID камеры в дереве настроек, которую необходимо сделать активной
"ARCH_FRAME_TIME" – поиск видеоархива по дате и времени	cam<>	-
	date<>	-
	time<>	-
	mode<>	Может принимать следующие значения: • 0 – видеошлюз, если задан (если не задан, то видеосервер) • 1 – видеосервер • 2 – долговременный архив • 10 + id объекта Внешнее хранилище на панели настройки объекта Монитор (в общем случае 11) – внешнее хранилище
"SETUP" – устанавливает параметры монитора	no_update<>	-
	overlay<>	Включение режима ускорения отображения
	x<>	Координата левого верхнего угла (0 – 100)
	y<>	Координата левого верхнего угла (0 – 100)
	w<>	Размер по горизонтали (0 – 100)
	h<>	Размер по вертикали (0 – 100)
	max_cams<>	Максимально допустимое число камер на мониторе
	min_cams<>	Минимально допустимое число камер на мониторе
	compress<>	-
	panel<>	Показать панель управления (0 – выключена, 1 – включена)
	panel_type<>	-
	s<>	-
	layout<>	-

Команда – описание команды	Параметры	Описание параметров
	gate<>	-
	map_id<>	-
	enable<>	-
	topmost<>	1 – показывать экран поверх всех остальных окон
	type<>	Тип объекта «Монитор»
	allow_move<>	Разрешить перемещение окна
	arch_id<>	Идентификатор архива
	cycle<>	Задержка при автоматическом листании (1 – 20 сек)
	flags<>	Флаги
	name<>	Имя объекта
	overlay<>	Включение режима ускорения отображения (0 – нет ускорения, 1 – ускорение «режим Оверлей», 2 – ускорение «режим DirectDraw»)
	tel_prior<>	Приоритет телеметрии
	gstream_version<>	Если значение не задано, функция автоматического выбора потока отключена. При значении параметра minBPS поток для отображения выбирается автоматически, как описано в разделе Настройка автоматического выбора видеопотока для отображения
"ACTIVATE" – активирование панели управления монитора	user_id<>	Идентификатор пользователя
	panel_active<>	-
"DEACTIVATE" – де активирование панели управления монитора	-	-
"EXPORT_FRAME" – экспорт кадра в JPG-файл	cam<>	-
	file	-
"KEY_PRESSED" – управление кнопками монитора видеонаблюдения и архива видеозаписей	number<>	-
	cam_id<>	ID камеры, к Окну видеонаблюдения которой требуется применить команду. Если идентификатор не указан, то команда применяется к активному Окну видеонаблюдения (см. Активное Окно видеонаблюдения)

Команда – описание команды	Параметры	Описание параметров
	key<>	Возможные значения: "ARCH_EDIT_DATE" – изменить дату поиска по архиву; "ARCH_EDIT_TIME" – изменить время поиска по архиву; "ARCH_EDIT_ENTER" – ввод изменений значений в архиве; "ARCH_EDIT_ESCAPE" – отменить редактирование архива; "ARCH_EDIT_BACK"; "ARCH_EDIT_REPLACE"; "WINDOW_ZOOM_IN" – развернуть окно видеонаблюдения; "WINDOW_ZOOM_OUT" – свернуть окно видеонаблюдения; "ZOOM_IN" – приближение изображения; "ZOOM_OUT" – отдаление изображения; "CYCLE_REW" – листание окон видеонаблюдения назад; "CYCLE_FF" – листание окон видеонаблюдения вперед; "LEFT" - сдвиг кадра влево в режиме Zoom; "RIGHT" – сдвиг кадра вправо в режиме Zoom; "UP" – сдвиг кадра вверх в режиме Zoom; "DOWN" – сдвиг кадра вниз в режиме Zoom; "MODE_VIDEO" – режим видеонаблюдения; "MODE_ARCH" – режим воспроизведения архивных видеозаписей; "MODE_ARCH2"- режим воспроизведения архивных видеозаписей 2; "MASK_SHOW" – нанести маску; "MASK_HIDE" – удалить маску; "ARM" – поставить камеру на охрану; "DISARM" – снять камеру с охраны; "REW" – обратная перемотка; "PLAY" – воспроизведение; "PLAY_NONSTOP" – безостановочное воспроизведение; "PLAY_FAST" – ускорить просмотр видеозаписи; "FF" – перемотка вперед; "RECORD" – запись; "RECORD_MIC" – запись с микрофона; "STOP" – остановка; "REC_STOP" – остановка записи; "PAUSE" – пауза; "MIC_ON" – микрофон включен; "MIC_OFF" – микрофон выключен; "PRINT" – вывод кадра на печать. "SELECT_LAYOUT" – управление раскладкой монитора видеонаблюдения; "START_CYCLE_FF" – включение функции автоматического листания окон видеонаблюдения вперед. Период листания задается при настройке интерфейсного объекта **Монитор** (см. Настройка режима отображения окон видеокамер) "STOP_CYCLE" – остановка автоматического листания Окон видеонаблюдения; "SCREEN.N" – выбор раскладки Окон видеонаблюдения. N принимает значения 1, 4, 6, 9, 16, 32 (максимальное значение зависит от количества Окон видеонаблюдения, добавленных в Монитор видеонаблюдения);

Команда – описание команды	Параметры	Описание параметров
		"EXPORT_DO" – запустить утилиту фонового экспорта AviExport (см. Утилита AviExport); "PROTECT_DO" – открыть окно создания закладки (см. Создание закладок); "PROTECT_VIEW" – открыть список закладок (см. Список закладок)
"START_AVI_EXPORT" – начать экспорт видео *Примечание. См. пример использования в* Примеры с Камерами и Монитором видеонаблюдения	start<>	Время начала
	finish<>	Время окончания
	avi_path<>	Путь к создаваемому файлу
	cam<>	ID камеры
"STOP_AVI_EXPORT" – остановить экспорт видео	monitor<>	Номер монитора
"START_AVI_SCHEDULE" – начать экспорт закладок	-	-
"STOP_AVI_SCHEDULE" – остановить экспорт закладок	-	-
"CONTROL_TELEMETRY" – Управление телеметрией См. также раздел Управление поворотными устройствами с помощью мыши	cam<>	ID камеры, на которой следует включить или отключить управление телеметрией при помощи мыши
	on<>	0 – отключить управление при помощи мыши 1 – включить управление при помощи мыши
"SET_REC_RESTART" – включить перезапуск записи при входе в архив		
"RESET_REC_RESTART" – отключить перезапуск записи при входе в архив		
"SET_ARCH_ENTER_PAUSE" – включить постановку проигрывания на паузу при входе в архив		
"RESET_ARCH_ENTER_PAUSE" – отключить постановку проигрывания на паузу при входе в архив		
"DISABLE_TELEMETRY" – отключить возможность управления телеметрией из Монитора видеонаблюдения		
"ENABLE_TELEMETRY" – включить возможность управления телеметрией из Монитора видеонаблюдения		
"INCREASE_VIEW" – увеличить размер окна камеры в Мониторе видеонаблюдения	cam<>	ID камеры
"DECREASE_VIEW" – уменьшить размер окна камеры в Мониторе видеонаблюдения	cam<>	ID камеры

Команда – описание команды	Параметры	Описание параметров
"SHOW_LAYOUT" – отобразить раскладку с указанным идентификатором	layout_id<>	Идентификатор раскладки в базе данных
"GO_LIVE" – переключить все камеры на мониторе в режим живого виде	-	-
"GO_ARCH" – переключить все камеры в мониторе в режим просмотра архива	arch_time<>	Необязательный параметр. Задает время позиционирования в архиве в формате ДД-ММ-ГГ ЧЧ:ММ:СС. По умолчанию архив позиционируется на последнюю запись
"SAVE_AS" – экспортировать выбранный фрагмент архива	-	-
"PLACE_CAM_IN_LAYOUT_CELL" – добавить камеру на Монитор в заданную ячейку заданной раскладки	cam<>	ID камеры в дереве объектов, которую необходимо вывести на монитор. Если значение параметра некорректно, например, 0 или -1, то соответствующая ячейка будет скрыта
	layout_name<>	ID или название раскладки, на которую необходимо добавить камеру
	cell<>	Номер ячейки на раскладке, в которую необходимо добавить камеру. Ячейки нумеруются сверху вниз и слева направо, начиная с верхнего левого угла раскладки. ***Внимание!*** *Нумерация ячеек начинается с 0.* Если в ячейку уже добавлена какая-либо другая камера, она будет заменена
"SET_TITLES" – показывать титры поверх видеоизображения в любом режиме отображения. Такие титры не записываются в архив и отображаются до применения команды CLEAR_TITLES либо перезагрузки Монитора	cam<>	ID камеры, к Окну видеонаблюдения которой требуется применить команду
	titles<>	Текст титров, который необходимо выводить. Для переноса строки используется '\r'
	title_id<>	ID титрователя
"CLEAR_TITLES" – выключить показ титров, созданных с помощью команды SET_TITLE	cam<>	ID камеры, к Окну видеонаблюдения которой требуется применить команду
	title_id<>	ID титрователя

Свойства объекта **MONITOR** показаны в таблице:

Свойства объекта MONITOR	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.4 PLAYER Аудиопроигрыватель

Объект **PLAYER** соответствует системному объекту **Аудиопроигрыватель**.

Список команд и параметров для объекта **PLAYER** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"PLAY_WAV" – проигрывает звуковой файл	file<>	Полный путь к звуковому файлу в формате .wav (с указанием имени проигрываемого файла, например: C:\Program Files (x86)\Intellect\Wav\cam_alarm_1.wav)

Команда – описание команды	Параметры	Описание параметров
	from_macro<ID>	Флаг реакции проигрывания звукового файла, отправленной с макрокоманды (с указанием ID существующей макрокоманды, ID > 0) *Примечание. Параметр не является обязательным, если канал голосового оповещения настроен в интерфейсе объекта **Аудиопроигрыватель** (см.* Настройка голосового оповещения с помощью объекта Аудиопроигрыватель*)*
"SETUP" – настройка параметров аудиопроигрывателя	board<>	Звуковое устройство проигрывателя архива
	flags<>	Флаги
	h<>	Высота диалога настройки (0 – 100)
	name<>	Имя объекта
	voice<>	Звуковое оповещение
	voice_board<>	Звуковое устройство оповещения
	w<>	Ширина диалога настройки (0 – 100)
	x<>	Левый верхний угол диалога настройки (0 – 100)
	y<>	Левый верхний угол диалога настройки (0 – 100)
"STOP_WAV" – завершение проигрывания файла	-	-

Свойства объекта **PLAYER** показаны в таблице:

Свойства объекта PLAYER	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.5 OLXA_LINE Микрофон

Объект **OLXA_LINE** соответствует системному объекту **Микрофон**.

От объекта **OLXA_LINE** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	Описание события
ACCU_START	Включение акустопуска
ACCU_STOP	Выключение акустопуска
ARM	Запись включена

Событие	Описание события
DISARM	Запись выключена
INCOMING_NUMBER	Входящий телефонный номер
OUTGOING_NUMBER	Исходящий телефонный номер
REC	Начало записи
REC_STOP	Конец записи
RESET	Подключение микрофона

Список команд и параметров для объекта **OLXA_LINE** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"ARM" – включить микрофон на запись	-	-
"DISARM" – выключить запись с микрофона	-	-
"SETUP" – настройка параметров микрофона	type<>	Тип линии
	accu_start<>	Порог срабатывания детектора звука
	accu_stop<>	Время удержания сработки детектора
	amp<>	Усиление
	aru<>	Автоматическая регулировка усиления
	aru_dyn<>	Уровень АРУ
	aru_time<>	Время срабатывания АРУ
	chan<>	Номер звукового канала микрофона
	compression<>	Тип компрессии
	flags<>	Флаги
	name<>	Имя объекта
	rec<>	Начало записи

Свойства объекта **OLXA_LINE** показаны в таблице:

Свойства объекта OLXA_LINE	Описание свойств объекта
ID<>	Идентификатор объекта

PARENT_ID<>	Идентификатор родительского объекта

Объект **OLXA_LINE** может находиться в состояниях, описанных в таблице:

Состояние объекта OLXA_LINE	**Описание состояния объекта**
"BLUE"	Микрофон снят с охраны
"GREEN"	Нет сигнала от микрофона
"YELLOW"	Микрофон поставлен на охрану
"RED"	Начало записи

7.6 DIALOG Окно запроса оператора

Объект **DIALOG** соответствует системному объекту **Окно запроса оператора**.

Список команд и параметров для объекта **DIALOG** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"SETUP" – настройка окна запроса оператора	x<>	Координата левого верхнего угла (0 - 100)
	y<>	Координата левого верхнего угла (0 - 100)
	allow_move<>	0 – запретить перемещение, 1 – разрешить перемещение
"RUN" – показать окно запроса оператора	-	-
"RUN_MODAL" – запуск окна запроса оператора в модальном режиме	-	-
"CLOSE" – закрывает последнее открытое окно запроса оператора	-	-
"CLOSE_ALL" – закрывает все открытые окна запроса оператора	-	-

7.7 MMS Сервис почтовых сообщений

Объект **MMS** соответствует системному объекту **Сервис почтовых сообщений**.

От объекта **MMS** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	**Описание события**
SET_CONNECTIONS	Список доступных подключений

Список команд и параметров для объекта **MMS** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"SETUP" – настройки для сервиса почтовых сообщений	smtp<>	Адрес SMTP сервера
	connection<>	Тип подключения
	smtp_username<>	Имя пользователя
	smtp_password<>	Пароль
	port<>	Номер порта
	flags<>	Флаги
	name <>	Название объекта
"GET_CONNECTIONS" – получить список доступных подключений	-	-

Свойства объекта **MMS** показаны в таблице:

Свойства объекта MMS	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.8 MAIL_MESSAGE Почтовое сообщение

Объект **MAIL_MESSAGE** соответствует системному объекту **Почтовое сообщение**.

От объекта **MAIL_MESSAGE** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	Описание события
SEND_ERROR	Ошибка отправки сообщения
SENT	Сообщение отправлено

Список команд и параметров для объекта **MAIL_MESSAGE** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"SETUP" – настройки для почтового сообщения	from<>	Адрес отправителя
	to<>	Адрес получателя
	cc<>	Копии
	subject<>	Тема сообщения

Команда – описание команды	Параметры	Описание параметров
	body<>	Тело сообщения
	attachments<>	Приложения. Если требуется приложить к письму несколько файлов, пути к файлам вложений разделяются точкой с запятой
	flags<>	Флаги
	name<>	Имя объекта
	pack<>	Способ упаковки приложений
	is_body_html<>	Указывает, применять ли HTML-разметку при отправке. Возможные значения: 1 и 0
	inline<>	Указывает, отображаются ли вложения только в тексте сообщения (значение 1) или и в тексте, и в разделе «Вложения» (значение 0)
"SEND" – отправка почтового сообщения	-	-
"SEND_RAW" – отправка почтового сообщения с заданием параметров	аналогично команде SETUP	См. пример в разделе Примеры скриптов на языке JScript

Свойства объекта **MAIL_MESSAGE** показаны в таблице:

Свойства объекта MAIL_MESSAGE	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.9 VMS Сервис голосовых сообщений

Объект **VMS** соответствует системному объекту **Сервис голосовых сообщений**.

Список команд и параметров для объекта **VMS** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"SEND" – послать сообщение	modem<>	Название устройства
	pulse<>	Тип набора (0 – тоновый, 1 – импульсный)
	name<>	Имя объекта
	redial_attempts<>	Количество попыток дозвона
	redial_delay<>	Пауза между попытками дозвона
	waitfordialtone<>	Ожидание сигнала линии (0 – нет, 1 – да)
	flags<>	Флаги

Свойства объекта **VMS** показаны в таблице:

Свойства объекта VMS	**Описание свойств объекта**
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.10 GRELE Реле

Объект **GRELE** соответствует системному объекту **Реле**.

От объекта **GRELE** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	**Описания события**
OFF	Реле выключено
ON	Реле включено
SIGNAL_LOST	Потеря связи

Список команд и параметров для объекта **GRELE** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"ON" – включить реле	-	-
"OFF" – выключить реле	-	-
"SETUP" – настройки для реле	chan<>	Номер выхода (0 – 15)
	flags<>	Флаги
	name<>	Имя объекта

Свойства объекта **GRELE** показаны в таблице:

Свойства объекта GRELE	**Описание свойств объекта**
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта
REGION_ID<>	Идентификатор региона

Объект **GRELE** может находиться в состояниях, описанных в таблице:

Состояние объекта GRELE	**Описание состояния объекта**
"ON"	Реле включено

Состояние объекта GRELE	Описание состояния объекта
"OFF"	Реле выключено
"DETACHED_ON"	Потеря связи
"DETACHED_OFF"	Потеря связи

7.11 GRAY Луч

Объект **GRAY** соответствует системному объекту **Луч**.

От объекта **GRAY** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	Описание события
ALARM	Тревога. Данное событие поступает при размыкании или замыкании луча (в зависимости от настройки объекта), если луч поставлен на охрану. Если луч снят с охраны, поступают события Луч разомкнут и Луч замкнут соответственно
ARM	Луч поставлен на охрану
CONFIRM	Тревога принята
DISARM	Луч снят с охраны
NOT_VALID_STATE	Зона не готова
OFF	Луч разомкнут. Данное событие поступает при размыкании луча, если луч снят с охраны
ON	Луч замкнут. Данное событие поступает при замыкании луча, если снят с охраны
SIGNAL_LOST	Потеря связи с лучом

Список команд и параметров для объекта **GRAY** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"ARM" – поставить на охрану луч	-	-
"DISARM" – снять с охраны луч	-	-
“CONFIRM” – принять тревогу	-	-
“SETUP” – настройки для луча	chan<>	Номер входа (0 – 15)
	flags<>	Флаги
	name<>	Имя объекта

Команда – описание команды	Параметры	Описание параметров
	type<>	Тип объекта луч (0 – на замыкание, 1 – на размыкание)

Свойства объекта **GRAY** показаны в таблице:

Свойства объекта GRAY	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта
REGION_ID<>	Идентификатор региона

Объект **GRAY** может находиться в состояниях, описанных в таблице:

Состояние объекта GRAY	Описание состояния объекта
"ARMED"	Луч поставлен на охрану
"DISARMED"	Луч снят с охраны
"ALARMED"	Тревога
"CONFIRMED"	Тревога принята
"DISARMED_ALARM"	Неготовность
"DETACHED_ARMED"	Потеря связи
"DETACHED_DISARM"	Потеря связи
“OFF”	Норма

7.12 VNS Сервис голосового оповещения

Объект **VNS** соответствует системному объекту **Сервис голосового оповещения**.

Список команд и параметров для объекта **VNS** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"SETUP" – настройка сервиса голосового оповещения	card<>	Имя звукового устройства. **Примечание**. Имя карты должно строго соответствовать тому названию, что указано в настройках звуковой карты **Сервиса голосового оповещения** системы *Интеллект*
	level<>	Уровень сигнала. Значение параметра варьируется от 0 до 15. По умолчанию оно равно 8, то есть среднему

Команда – описание команды	Параметры	Описание параметров
	channel<>	Набор звуковых каналов. Возможные значения параметра: 0 – нет звукового канала; 1 – левый канал воспроизведения; 2 – правый канал воспроизведения; 3 – левый и правый канал воспроизведения (оба канала)
	flags<>	Флаги
	ip<>	IP-адрес сетевого устройства
	name<>	Имя объекта
	pass<>	Пароль
	user<>	Имя пользователя
"PLAY" – проигрывание звукового файла	file<>	Полный путь к звуковому файлу в формате .wav (с указанием имени проигрываемого файла, например: C:\Program Files (x86)\Intellect\Wav\cam_alarm_1.wav). **Примечание**. Если указано только имя файла, то путь к нему по умолчанию будет взят с ключа реестра InstallPath в разделе «HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect» (HKEY_LOCAL_MACHINE\Software\Wow6432Node\ITV\Intellect для 64-битной системы), в значении параметра «InstallPath». Также в данном параметре есть возможность указать проигрывание нескольких музыкальных файлов с помощью операции «+»
"STOP" – завершение проигрывания файла	-	-

Свойства объекта **VNS** показаны в таблице:

Свойства объекта VNS	Описание свойства объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.13 SMS Сервис коротких сообщений

Объект **SMS** соответствует системному объекту **Сервис коротких сообщений**.

От объекта **SMS** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события	Комментарий
RECEIVE	Получено сообщение	Если событие не поступает при получении сообщения на модем, следует использовать ключ реестра ProcessFromSim (см. Справочник ключей реестра). В параметре сообщения message<> содержится текст присланного сообщения. В параметре phone<> содержится телефон, с которого поступило сообщение, в формате +7XXXXXXXXXX

Список команд и параметров для объекта **SMS** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"SETUP" – настройка сервиса коротких сообщений	device<>	SMS устройство
	flags<>	Флаги
	message<>	Текст сообщения
	name<>	Имя объекта
	phone<>	Номер телефона

Свойства объекта **SMS** показаны в таблице:

Свойства объекта SMS	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.14 TELEMETRY Поворотное устройство

Объект **TELEMETRY** соответствует системному объекту **Поворотное устройство**.

От объекта **TELEMETRY** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события	Комментарий
LOCKED	Заблокировано	Событие поступает после команды LOCK (см. таблицу ниже)
UNLOCKED	Разблокировано	Событие поступает после команды UNLOCK (см. таблицу ниже)

Список команд и параметров для объекта **TELEMETRY** представлен в таблице.

Команда – описание команды	Параметры	Описание параметров
"AUTOFOCUS_ON" – включить функцию автонаведения (автофокус)	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"AUTOPAN_END_P" – задать конечную точку автоповорота	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"AUTOPAN_START" – начать автоповорот	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"AUTOPAN_START_P" – задать стартовую точку автоповорота	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"AUTOPAN_STOP" – окончить автоповорот	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"CLEAR_PRESET" – очистить выбранный пресет	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
	preset<>	Пресет

Команда – описание команды	**Параметры**	**Описание параметров**
"D2OFF" – отключить дополнительные динамические настройки для поворотных видеокамер Panasonic, предназначенные для улучшения качества аналогового видеосигнала	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"D2ON" – включить дополнительные динамические настройки для поворотных видеокамер Panasonic, предназначенные для улучшения качества аналогового видеосигнала	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"DOWN" – повернуть объектив видеокамеры вниз	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"FOCUS_IN" – увеличить изображение	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"FOCUS_OUT" – уменьшить изображение	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"FOCUS_STOP" – остановить увеличение/уменьшение изображения	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"GO_PRESET" – повернуть видеокамеру в положение, заданное на пресете	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
	preset<>	Пресет
"HOME" – повернуть видеокамеру в исходную (домашнюю) позицию	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"IRIS_CLOSE" – закрыть диафрагму	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"IRIS_OPEN" – открыть диафрагму	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"IRIS_STOP" – остановить диафрагму	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"LEFT" – повернуть объектив видеокамеры влево	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"LEFT_DOWN" – повернуть объектив видеокамеры влево и вниз	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"LEFT_UP" – повернуть объектив видеокамеры влево и вверх	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"PATROL_LEARN" – начать процедуру программирования патрулирования, выполняемую путем записи поведения видеокамеры	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
	point<>	Номер точки
	preset<>	Номер пресета (тура)
	dwell<>	Время нахождения в точке в секундах
	speed<>	Скорость перемещения в точку
	flush_tour<>	1 – записать тур 0 – не записывать тур

Команда – описание команды	**Параметры**	**Описание параметров**
"PATROL_PLAY" – начать патрулирование	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"PATROL_STOP" – закончить патрулирование	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"RIGHT" – повернуть объектив видеокамеры вправо	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"RIGHT_DOWN" – повернуть объектив видеокамеры вправо и вниз	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"RIGHT_UP" – повернуть объектив видеокамеры вправо и вверх	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"SET_PRESET" – записать текущее положение видеокамеры в выбранный пресет	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
	preset<>	Пресет
"STOP" – завершить поворот объектива видеокамеры	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"UP" – повернуть объектив видеокамеры вверх	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий)
"SETUP" – настроить поворотное устройство	address<>	Адрес устройства
	cam<>	Идентификатор камеры для управления
	flags<>	Флаг работы объекта (0 – включен, 1 – отключен)
	name<>	Имя объекта поворотного устройства
	speed<>	Скорость
"SEND_BUFFER" – отправить команду в шестнадцатеричном формате в COM-порт	buffer<>	Команда в шестнадцатеричном формате
	parent_id<>	Номер родительского объекта **Контроллер телеметрии.** Обязательный параметр
	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий). Значение параметра должно быть больше 0
"LOCK" – заблокировать. Перевод телеметрии в состояние LOCKED на заданное время	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий). Значение параметра должно быть больше 0. На время блокировки запрещается выполнение команд управления с более низким приоритетом, чем указанный
	duration<>	Длительность наложения блокировки. Если параметр не указан, блокировка действует до выполнения команды UNLOCK
"UNLOCK" – разблокировать. Перевод телеметрии в состояние UNLOCKED	-	-

Команда – описание команды	Параметры	Описание параметров
"AUTOFOCUS_OFF" – выключить функцию автонаведения (автофокус)	tel_prior<>	Приоритет (1 – низкий, 2 – средний, 3 – высокий). ⚠ Для использования этой команды её необходимо добавить на вкладку **Реакции** для объекта **TELEMETRY** в ddi.exe (см. Закладка Реакции)

Свойства объекта **TELEMETRY** показаны в таблице:

Свойства объекта TELEMETRY	Описание свойств объекта
ID<>	Идентификатор объекта поворотного устройства
PARENT_ID<>	Идентификатор родительского объекта

Объект **TELEMETRY** может находиться в состояниях, описанных в таблице:

Состояние объекта TELEMETRY	Описание состояния объекта
LOCKED – Заблокировано	Управление телеметрией заблокировано с некоторым приоритетом. Запрещено управление телеметрией с приоритетом ниже указанного при блокировке (см. таблицу выше)
UNLOCKED – Разблокировано	Разрешено управление телеметрией с любым приоритетом

7.15 TELEMETRY_EXT Пульт управления

Объект **TELEMETRY_EXT** соответствует системному объекту **Пульт управления**.

От объекта **TELEMETRY_EXT** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события	Параметр	Описание параметра	Диапазон значений
KEY_PRESSED	Нажата клавиша	param0<>	Код нажатой клавиши	См. Руководство по установке и настройке компонентов охранной системы
		device<>	Устройство, на котором нажата клавиша	0 – Основная клавиатура *AXIS T8312* 1 – Клавиатура *AXIS T8313*
KEY_RELEASED	Отпущена клавиша	param0<>	Код отпущенной клавиши	0..21 для *AXIS T8312*. Для *BOSCH KBD-Digital, BOSCH KBD-Universal* и *Panasonic WV-CU950* см. Руководство по установке и настройке компонентов охранной системы
		device<>	Устройство, на котором отпущена клавиша	0 – Основная клавиатура *AXIS T8312* 1 – Поворотный переключатель *AXIS T8313*
MOVED	Изменено положение	param0<>	Значение смещения	Для колеса поворотного переключателя JogDial -1.. 1; для колеса покадровой прокрутки Shuttle -7..7 Для пульта *Panasonic WV-CU950* JogDial -1.. 1; Shuttle -6..6

Событие	Описание события	Параметр	Описание параметра	Диапазон значений
		device<>	Тип использованного механизма управления *AXIS T8313*	0 – колесо поворотного переключателя 1 – колесо покадровой прокрутки

Список команд и параметров для объекта **TELEMETRY_EXT** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"DRAW_FIGURE" – нарисовать фигуру на дисплее пульта телеметрии *BOSCH KBD-Digital или BOSCH KBD-Universal*	display<>	0x00 – основной дисплей, 0x01 – статусный дисплей
	x1<>	Начальная координата по оси X (от 0 до 127 для основного дисплея, от 0 до 121 для статусного)
	y1<>	Начальная координата по оси Y (от 0 до 239 для основного дисплея, от 0 до 31 для статусного)
	x2<>	Конечная координата по оси X (от 0 до 127 для основного дисплея, от 0 до 121 для статусного)
	y2<>	Конечная координата по оси Y (от 0 до 239 для основного дисплея, от 0 до 31 для статусного)
	is_fill<>	0 – не закрашивать фигуру, 1 – закрашивать фигуру
	is_set_pixels<>	0 – стереть фигуру с дисплея, 1 – нарисовать фигуру
	figure<>	0 – линия, 1 – прямоугольник
"PRINT_TEXT" – напечатать текст на дисплее пульта телеметрии *BOSCH KBD-Digital или BOSCH KBD-Universal*	display<>	0x00 – основной дисплей, 0x01 – статусный дисплей
	x<>	Координата по оси X (от 0 до 127 для основного дисплея, от 0 до 121 для статусного)
	y<>	Координата по оси Y (от 0 до 239 для основного дисплея, от 0 до 31 для статусного)
	charset<>	Кодировка: 0 – Латинская 1 – Кириллическая 2 – Центральноевропейская
	style<>	Стиль: 0 – Обычный 1 – Полужирный
	text<>	Текстовое сообщение
"PRINT_TEXT" – напечатать текст на дисплее пульта телеметрии *Panasonic WV-CU950*	y<>	0 – вывести текст на первую строку 1 – вывести текст на вторую строку

Команда – описание команды	**Параметры**	**Описание параметров**
	text<>	Выводимый текст строки, максимум 20 символов
	flickering<>	Строка из шести символов, определяющая параметры мигания текста: d1 d2 d3 d4 d5 d6 d1 определяет период мигания: 0 – мигание отключено 1 – период 0.25 сек, символ заменяется белым пробелом 2 – период 0.5 сек, символ заменяется белым пробелом 3 – период 0.75 сек, символ заменяется белым пробелом 4 – период 1 сек, символ заменяется белым пробелом 5 – период 0.25 сек, символ заменяется темным пробелом 6 – период 0.5 сек, символ заменяется темным пробелом 7 – период 0.75 сек, символ заменяется темным пробелом 8 – период 1 сек, символ заменяется темным пробелом d2: 1 – мигают символы с 1 по 4, 0 – данные символы не мигают d3: 1 – мигают символы с 5 по 8, 0 – данные символы не мигают d4: 1 – мигают символы с 9 по 12, 0 – данные символы не мигают d5: 1 – мигают символы с 13 по 16, 0 – данные символы не мигают d6: 1 – мигают символы с 17 по 20, 0 – данные символы не мигают
"CLEAR_DISPLAY" – очистить дисплей пульта телеметрии *BOSCH KBD-Digital или BOSCH KBD-Universal.* Для пульта телеметрии *Panasonic WV-CU950* реакция без параметров	display<>	0x00 – основной дисплей, 0x01 – статусный дисплей
"RELE_ON" – включить лампочку на клавиатуре *AXIS T8312* или пульте *Panasonic WV-CU950*	rele<>	Код клавиши с лампочкой, 12..16 для *AXIS T8312*. Для Panasonic WV-CU950 см. Руководство по установке и настройке компонентов охранной системы, раздел Особенности настройки и работы с пультом управления телеметрией Panasonic WV-CU950
"RELE_OFF" – выключить лампочку на клавиатуре *AXIS T8312* или пульте *Panasonic WV-CU950*	rele<>	Код клавиши с лампочкой, 12..16
"RESET" – физическая перезагрузка пульта *Panasonic WV-CU950*	type<>	0 – немедленная перезагрузка 1 – перезагрузка по истечении 100мсек 2 – перезагрузка по истечении 200мсек 3 – перезагрузка по истечении 500мсек 4 – перезагрузка по истечении 1сек

Команда – описание команды	Параметры	Описание параметров
"SET_ALARM" – задает вид тревожного сигнала пульта *Panasonic WV-CU950*	audio_alarm<>	0 – звук отключен 1 – простой однократный сигнал тревоги 2 – простой двукратный сигнал тревоги 3 – простой троекратный сигнал тревоги 4 – однократный сигнал тревоги длительностью 0.1 сек 5 – однократный сигнал тревоги длительностью 0.2 сек 6 – однократный сигнал тревоги длительностью 0.3 сек 7 – однократный сигнал тревоги длительностью 1 сек 8 – простое однократное звучание 9 – простое двукратное звучание A – простое троекратное звучание B – однократный сигнал длительностью 0.1 сек C – однократный сигнал длительностью 0.2 сек D – однократный сигнал длительностью 0.3 сек E – однократный сигнал длительностью 1 сек F – сигнал тревоги

7.16 MACRO Макрокоманда

Объект **MACRO** соответствует системному объекту **Макрокоманда**.

От объекта **MACRO** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	Описание событий	Параметры	Описание параметров
RUN	Выполнено действие	src_sender<>	Имя компьютера, на котором была запущена макрокоманда. *Примечание. Значение данного параметра будет отображаться в Протоколе событий в столбце **Доп. инфо** в реальном времени. При перезапуске ПК Интеллект и загрузке записей протокола событий из БД данная информация не отображается в интерфейсе, но остается в базе данных*
		user_id<>	Идентификатор пользователя, выполнившего макрокоманду. *Примечание. Значение данного параметра вместе с именем пользователя будет отображаться в Протоколе событий в столбце **Доп. инфо** в реальном времени. При перезапуске ПК Интеллект и загрузке записей протокола событий из БД данная информация не отображается в интерфейсе, но остается в базе данных*

Список команд и параметров для объекта **MACRO** представлен в таблице:

Свойства объекта **MACRO** показаны в таблице:

Свойства объекта MACRO	Описание свойств объекта
ID<>	Идентификатор объекта

PARENT_ID<>	Идентификатор родительского объекта

Объект **MACRO** может находиться в состояниях, описанных в таблице:

Состояние объекта MACRO	**Описание состояния объекта**
"NORM"	Норма

7.17 TIME_ZONE Временная зона

Объект **TIME_ZONE** соответствует системному объекту **Временная зона**.

От объекта **TIME_ZONE** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	**Описание событий**
ACTIVATE	Начало
DEACTIVATE	Конец

Список команд и параметров для объекта **TIME_ZONE** представлен в таблице:

Свойства объекта **TIME_ZONE** показаны в таблице:

Свойства объекта TIME_ZONE	**Описание свойств объекта**
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

Объект **TIME_ZONE** может находиться в состояниях, описанных в таблице:

Состояние объекта TIME_ZONE	**Описание состояния объекта**
"ACTIVE"	Активный
"INACTIVE"	Неактивный

7.18 SSS_WATCHDOG Служба перезагрузки системы

Объект **SSS_WATCHDOG** соответствует системному объекту **Служба перезагрузки системы**.

От объекта **SSS_WATCHDOG** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	**Описание событий**
RESTART_EXCEEDED	Превышено количество перезагрузок модуля
RESTART_PROCESS	Перезагрузка модуля

Список команд и параметров для объекта **SSS_WATCHDOG** представлен в таблице:

Свойства объекта **SSS_WATCHDOG** показаны в таблице:

Свойства объекта SSS_WATCHDOG	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта

7.19 SLAVE Компьютер

Объект **SLAVE** соответствует системному объекту **Компьютер**.

От объекта **SLAVE** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	Описание событий	Комментарий
CONNECTED	Подключение	Событие генерируется, когда какой-либо Клиент подключился к Серверу
DISCONNECTED	Отключение	Событие генерируется, когда какой-либо Клиент отключился от Сервера
KEY_IGNORED_HW	Ключ отвергнут (несоответствие кодов плат)	Событие генерируется в случае, если коды плат (либо HID) в ключе не соответствуют текущим у компьютера
KEY_IGNORED_SW	Ключ отвергнут (превышено ограничение)	Событие генерируется при наличии софтверных ограничений. Например, когда ключ подходит, но количество созданных в дереве оборудования объектов больше, чем указано в ключе
KEY_UPDATED	Ключ обновлен	
PROTOCOL_RCVD	Протокол получен	
REBUILD_IN_START	Начало переиндексации архива	
REBUILD_IN_STOP	Окончание переиндексации архива	
REGISTER_ATTEMPT	Попытка несанкционированного доступа	
REGISTER_ERROR	Превышен лимит попыток доступа	Событие генерируется, когда пользователь много раз предпринимал неудачные попытки входа в систему. После события возникает некоторый таймаут, когда данный пользователь не сможет сделать попытку входа. Количество попыток входа и таймаут можно изменить через реестр
REGISTER_USER	Регистрация пользователя	Данное событие происходит при попытке пользователя войти в систему (при вводе логина и пароля)
DISC_EXIST	Диск для записи архива присутствует	
NO_DISC	Диск для записи архива отсутствует	
KEY_IGNORED_FR	Ключ отвергнут	Событие генерируется в случае, если ключевой файл не удалось записать на диск

События	**Описание событий**	**Комментарий**
SHUTDOWN	Завершение работы	
DISC_MOUNT	Диск подключен (монтирован)	
DISC_UNMOUNT	Диск отсоединен (размонтирован)	
ARCHIVE_DEPTH	Глубина архива	Событие генерируется в полночь и содержит информацию о глубине архива по всем дискам в часах (параметр depth<>). Для вызова события вручную используется реакция GET_DEPTH. При отображении события в Протоколе событий в поле **Дополнительная информация** указывается глубина архива в формате Дни:Часы. Также данная информация содержится в параметре события param0<>. Глубина архива рассчитывается как разница между датами создания самого старого файла архива и самого нового файла архива (на диске или по камере)
FORCED_OFF	Принудительная выгрузка	Событие генерируется перед принудительной выгрузкой ПК *Интеллект*, например, в случае, если извлечен ключ защиты Guardant. Выгрузка производится после повлекшего ее действия (например, извлечения ключа Guardant) через интервал времени, задаваемый ключом реестра UnloadDelay – см. Справочник ключей реестра
DEACTIVATE_ALL_DISP	Скрыть все экраны	Событие позволяет скрыть все экраны на указанном в параметре slave<> компьютере. Если в событии присутствует параметр except<>, то скрываются все экраны, кроме экрана с указанным в данном параметре идентификатором
LIC_EXPIRATION	Действие лицензии заканчивается через	По умолчанию не генерируется. Для включения необходимо установить ключ NotifyExpireLic = 1 (см. Справочник ключей реестра). В параметре days<> указывается количество дней до окончания лицензии (может быть дробным числом). Событие генерируется в момент загрузки ПК *Интеллект* и после смены дня
DATABASE_ERROR	Потеряна связь с базой данных	Событие генерируется в случае разрыва связи с SQL Server при первом обращении к нему после разрыва
SCRIPT_ERROR	Выполнение скрипта завершилось ошибкой	По умолчанию событие не генерируется, так как не добавлено в файл внешних настроек intellect.ddi. Чтобы событие SCRIPT_ERROR генерировалось и записывалось в таблицу PROTOCOL, его необходимо добавить в таблицу intellect.ddi (см. Редактирование файла внешних настроек intellect.ddi с помощью утилиты ddi.exe)

Список команд и параметров для объекта **SLAVE** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"**SETUP**" – установить параметры для компьютера	display_id<>	Идентификатор экрана
	drives<>	Диски для записи видеоархива
	drives_a<>	Диски для записи аудиоинформации

Команда – описание команды	Параметры	Описание параметров
	flags<>	Флаги
	arch_days<>	Размер архива событий
	connection<>	Соединение
	disable_protocol<>	Отключить протоколирование
	ip_address<>	IP адрес устройства
	is_backup<>	Архивация
	is_load<>	Загружен
	local_protocol<>	Локальный протокол
	modem<>	Модемное соединение
	name<>	Имя объекта
	password<>	Пароль
	sync_time<>	Синхронизировать время
	username<>	Имя пользователя
"BACKUP" – сделать резервную копию БД	-	-
"CONNECT_ONE" – подключиться к компьютеру. Подключает соответствующий компьютер. Следует избегать использования этой реакции вручную	-	-
"CONNECT_OTHER" – подключиться к ядрам. Подключает компьютер к другим ядрам из конфигурации. Следует избегать использования этой реакции вручную	-	-
"DISCONNECT_ONE" – отключиться от компьютера. Отключает соответствующий компьютер. В случае отключения ядро может автоматически подключиться. Следует избегать использования этой реакции вручную	-	-
"SYNC_PROTOCOL" – запустить утилиту синхронизации протокола SyncProtocol.exe. Если синхронизация настроена, происходит слияние протокола	-	-
"SYNC_TIME" – синхронизовать время. Для выполнения данной реакции необходимо, чтобы в разделе реестра HKEY_LOCAL_MACHINE\SOFTWARE\ITV\INTELLECT\ (HKEY_LOCAL_MACHINE \Software\Wow6432Node\ITV\INTELLECT для 64-битной системы) был создан параметр SyncTime со значением 1 на той системе, которой адресована реакция	-	-

Команда – описание команды	Параметры	Описание параметров
"CREATE_PROCESS" – запустить процесс	command_line<>	Командная строка. Команды командной строки Windows, записанные без символов переноса строки через разделители |, & или &&
"SEND_MY_CONFIG" – разослать конфигурацию. Рассылает свою конфигурацию другим компьютерам. То же, что “SPREAD_CONFIG”	-	-
"MOVE_CONFIG" – переместить конфигурацию. Перемещает конфигурацию, созданную в дереве объектов на основе компьютера-Поставщика, на компьютер-Получатель	from<>	Поставщик
	to<>	Получатель
"SPREAD_CONFIG" – распространить конфигурацию, то же, что “SEND_MY_CONFIG”	-	-
"GET_DEPTH" – получить глубину архива. В ответ на реакцию в системе формируется событие ARCHIVE_DEPTH (см. таблицу выше). Отсутствие одного или обоих параметров означает запрос глубины по записям для всех возможных значений параметра	cam<>	Идентификатор камеры, для которой запрашивается глубина архива
	drive<>	Диск или сетевой путь, по которому запрашивается глубина архива. Название диска задается формате "<буква диска>:\\", например drive<D:\\> *Примечание. Символ "\" – экранируемый.* Сетевой путь задается в формате UNC
"ACTIVATE_DISPLAY" – сменить экран. Команда позволяет отобразить на мониторе (мониторах) компьютера Экран с заданным идентификатором	display_id<>	Идентификатор соответствующего объекта **Экран**. Если в параметре передано пустое значение, при выполнении данной команды скрываются все экраны

Свойства объекта **SLAVE** показаны в таблице:

Свойства объекта SLAVE	Описание свойств объекта
ID<>	Идентификатор объекта
PARENT_ID<>	Идентификатор родительского объекта
USER_ID<>	Идентификатор пользователя

7.20 DISPLAY Экран

Объект **DISPLAY** соответствует системному объекту **Экран**.

От объекта **DISPLAY** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события
ACTIVATE	Активация экрана
DEACTIVATE	Деактивация экрана

ACTIVATED	Активация экрана на удаленном компьютере. В параметре param0<> передается имя компьютера

Список команд и параметров для объекта **DISPLAY** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"ACTIVATE" – показать экран	macro_slave_id<>	Имя компьютера, на котором должен быть показан экран
"DEACTIVATE" – скрыть экран	macro_slave_id<>	Имя компьютера, на котором должен быть скрыт экран

Примечание.

Если параметр «macro_slave_id» не установлен, то команда будет выполнена для всех компьютеров в системе.

Свойства объекта **DISPLAY** показаны в таблице:

Свойства объекта DISPLAY	**Описание свойств объекта**
flags	Флаги
id	Идентификатор объекта
name	Имя объекта
parent_id	Идентификатор родительского объекта

7.21 GATE Видеошлюз

Объект GATE соответствует системному объекту **Видеошлюз.**

От объекта **GATE** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	**Описание событий**	**Комментарий**
GATE_LOW_FPS	Упал темп ввода на шлюзе	
ACTIVE	Шлюз активен	Событие генерируется, когда список работающих камер соответствует списку конфигурации Видеошлюза
INACTIVE	Шлюз неактивен	Событие генерируется, когда нет запроса потоков видео через Видеошлюз
ACTIVE_PART	Частичная работа шлюза	Событие генерируется, когда количество реально работающих камер меньше, чем в списке Видеошлюза

Список команд и параметров для объекта **GATE** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров	Особенности
"START_VIDEO" – включение видеопотока камеры и запуск записи в архив	cam<>	Идентификатор камеры, по которой нужно запустить или остановить запись	Команды работают, даже если в Мониторе не отображается выбранная камера. Команды работают, если в Видеошлюзе включена постоянная запись и запись по активной камере – см. Настройка записи в архив Видеошлюза
"STOP_VIDEO" – остановка видеопотока камеры и записи в архив			

7.22 CAM_VMDA_DETECTOR Детектор VMDA

Объект **CAM_VMDA_DETECTOR** соответствует системному объекту **Детектор VMDA**.

От объекта **CAM_VMDA_DETECTOR** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	Описание события	Параметр	Описание параметра
ALARM	Тревога	native_type<>	Для включения данного параметра необходимо задать следующие ключи реестра: VMDA.determineNoise, VMDA.determineGivenTaken, VMDA.determineHumanCar (см. Справочник ключей реестра). Параметр принимает значения: -1 – неизвестный тип объекта (начальное состояние) 0 – другое 1 – человек или группа людей (в зависимости от значения параметра native_value<>: если 1, то человек, если >1, то группа людей) 2 – машина 3 – шум 4 – принесенный предмет 5 – унесенный предмет
		native_value<>	Для включения данного параметра необходимо задать следующие ключи реестра: VMDA.determineNoise, VMDA.determineGivenTaken, VMDA.determineHumanCar (см. Справочник ключей реестра). Счетчик людей для объекта типа «человек». Позволяет определить число людей в группе. Для остальных типов объектов равен -1
		param0<>	Строковое значение, содержащее параметры события от детектора VMDA
ALARM_END	Конец тревоги		
ARMED	Детектор VMDA поставлен на охрану		
DISARMED	Детектор VMDA снят с охраны		

Список команд и параметров для объекта **CAM_VMDA_DETECTOR** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"ARM" – поставить на охрану детектор	-	-
"DISARM" – снять с охраны детектор	-	-

Примечание

Если видеокамеры подключены с помощью ONVIF-Сервера, то события от **Детектора VMDA** и прочих интеллектуальных детекторов будут передаваться как события от встроенных детекторов – см. CAM_IP_DETECTOR Детектор встроенный.

7.23 ARCH Долговременный архив

Объект ARCH соответствует системному объекту **Долговременный архив.**

От объекта ARCH поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

События	Описание событий	Комментарий
ACTIVE	Долговременный архив активен	Событие генерируется, когда список камер, видео с которых архивируется, соответствует списку конфигурации Долговременного архива
INACTIVE	Долговременный архив неактивен	Событие генерируется, когда не производится архивация через Долговременный архив
ACTIVE_PART	Частичная работа Долговременного архива	Событие генерируется, когда включена архивация видео не от всех камер, указанных в списке Долговременного архива

7.24 CORE Ядро

Объект **CORE** – это ядро системы, глобальный статический объект, реализующий методы, используемые для контроля состояния и управления системными объектами программного комплекса *Интеллект*. Более широкие возможности для работы с объектом CORE предоставляются при использовании скриптов на языке программирования JScript – см. документ Руководство по программированию (JScript).

От объекта CORE поступают события, представленные в таблице:

Событие	Описание события
DO_REACT	Событие инициирует реакцию того или иного объекта в системе. В параметре action данного события передается описание действия, которое требуется выполнить. Примеры значений параметра action: SET_MARKRECT – посылается при обнаружении лица на видеоизображении; DEL_MARKRECT – посылается при исчезновении лица с видеоизображения. Также могут присутствовать другие параметры события, которые можно отследить при помощи Отладочного окна (см. Отладочное окно). Например, если значение параметра action равно SET_MARKRECT, то в параметре param5_val передается номер камеры, на видеоизображении с которой обнаружено лицо. Об этом говорит название параметра, передаваемое в параметре param5_name. Для значения DEL_MARKRECT номер камеры передается в параметре param0_val

SLAVE_CHANGED	Событие генерируется при срабатывании Сервиса отказоустойчивости (Failover). Содержит следующие параметры: • old_slave_id – идентификатор объекта **Компьютер**, с которого переносятся камеры • new_slave_id – идентификатор объекта **Компьютер**, на который переносятся камеры • CAM<n1, n2, ... > – где n1, n2 и т.д. являются идентификаторами камер, перенесенных под другой родительский объект **Компьютер**. Например, CAM<4,6,7> – перенос камер с идентификаторами 4, 6, 7
CREATE_OBJECT	Событие инициирует создание объекта. Параметры: • objtype<> – тип объекта, например, objtype<PERSON> – создание пользователя • parent_id<> – идентификатор родительского объекта • service_photo<> – при создании пользователя в данном параметре передается кодированное в base64 бинарное изображение – фотография пользователя. Параметр необходим, чтобы при создании пользователя в Бюро пропусков ему можно было сразу назначить фотографию

7.25 TITLEVIEWER Поиск по титрам

Объект **TITLEVIEWER** соответствует системному объекту **Поиск по титрам**.

От объекта **TITLEVIEWER** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

Событие	**Описание события**	**Параметры**	**Описание параметров**	**Комментарий**
GO_VIDEO	Запрос видео	<cam>	Идентификатор камеры, по которой найдены титры	Событие генерируется при двойном клике левой кнопкой мыши на строке, содержащей результат поиска
		<date>	Дата	
		<time>	Время	

7.26 MAP Карта

Объект **MAP** соответствует системному объекту **Карта**.

От объекта **MAP** поступают события, представленные в таблице ниже. Запуск процедуры происходит при возникновении соответствующего события.

Событие	**Описание события**
LAYER_ACTIVATED	Активация слоя. Данное событие поступает при переходе на слой карты. В параметре obj_id<> содержится идентификатор активированного слоя
ACTIVATE_OBJECT	Активация объекта. Событие поступает при выборе (активации кликом мыши) объекта на карте. Параметры: 1. obj_type<> – тип объекта 2. user_id<> – идентификатор пользователя 3. module<> – имя модуля, для Карты – map.run 4. date<> – дата возникновения события 5. time<> – время возникновения события 6. slave_id<> – сетевое имя компьютера

	7. obj_id<> – идентификатор объекта 8. layer<> – идентификатор слоя Карты 9. fraction<> – миллисекунда, в которую событие возникло 10. owner<> – пользователь, активировавший объект 11. type_of_display<> – тип отображения объекта, возможные значения: a. IMAGE – изображение b. IMAGE_AND_INDICATOR – изображение и индикатор c. TEXT – текст d. LINE – линия e. POLYGON – многоугольник f. ELIPSIS – эллипс g. TITLE – название объекта
OBJDBLCLK	Событие поступает при двойном клике по объекту на Карте. Содержит те же параметры, что ACTIVATE_OBJECT

Список команд и параметров для объекта **MAP** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"SET_TOPMOST" – поверх всех окон	-	-
"SET_NOTOPMOST" – отмена поверх всех окон	-	-
"HIDE_OBJECT" – скрыть/показать значки объектов на карте	objtype<>	Тип объекта. Может быть пустым. Если тип объекта не задан, скрываются/отображаются объекты всех типов
	objid<>	Идентификатор объекта. Может быть пустым. Если идентификатор объекта не задан, скрываются/ отображаются все объекты заданного типа
	hide<>	0 – объекты отображаются на карте 1 – объекты не отображаются на карте
"SET_OBJECT_GEOMETRY" – задать положение объекта на карте	objtype<>	Тип объекта
	objid<>	Идентификатор объекта
	x<>	Новая координата верхнего левого угла значка объекта на слое карты в пикселях по оси X
	y<>	Новая координата верхнего левого угла значка объекта на слое карты в пикселях по оси Y
	exclude_children<>	По умолчанию при использовании реакции SET_OBJECT_GEOMETRY при перемещении значков объектов перемещаются и названия этих объектов (дочерние объекты). Если передать в реакции параметр exclude_children<1>, то объект перемещается отдельно от дочерних, то есть без названия
"INSCRIBE" – вписать в окно	-	-
"SHOW_MINIMAP" – показать миникарту	x<>	Координата верхнего левого угла миникарты по оси X в пикселях
	y<>	Координата верхнего левого угла миникарты по оси Y в пикселях
	w<>	Ширина миникарты в пикселях
	h<>	Высота миникарты в пикселях

	monitor<>	Идентификатор монитора
	slave_id<>	Сетевое имя компьютера
"SET_ZOOM" – изменить масштаб карты	zoom<>	Задаваемый масштаб карты
"ACTIVATE_OBJECT" – активировать объект на карте	obj_type<>	Тип объекта
	obj_id<>	Идентификатор объекта
	layer<>	Идентификатор слоя карты. Если параметр указан, то скрипт будет работать на указанном слое, если не указан, то на текущем слое
"DRAW_ARROW" – нарисовать трек перемещения между объектами	first_obj_type<>	Тип объекта, от которого будет строиться трек
	first_obj_id<>	Идентификатор объекта, от которого будет строиться трек
	second_obj_type<>	Тип объекта, к которому будет строиться трек
	second_obj_id<>	Идентификатор объекта, к которому будет строиться трек
	obj_id<>	Идентификатор создаваемого трека
"ERASE_ARROW" – удалить трек перемещения между объектами	obj_id<>	Идентификатор трека, который надо удалить. Если не указывать параметр, то будут удалены все треки

Особенности реализации команды **DRAW_ARROW**:

1. В результате запуска команды трек будет отображаться на каждом слое **Карты** в виде стрелок  между заданными объектами.
2. Если объекты расположены на одном слое, то стрелка рисуется напрямую между указанными объектами. Если объекты на разных слоях, то стрелка рисуется по самому короткому пути.
3. Можно ограничить глубину поиска взаимосвязей по слоям для построения трека ключом DrowArrowSearchDepth, см. Справочник ключей реестра.
4. Если
   a. невозможно построить трек,
   b. одного из объектов не существует,
   c. можно построить трек, но невозможно отобразить стрелки,

   то на стартовом объекте будет отображаться  , на конечном 

7.27 FAILOVER Сервис отказоустойчивости

Объект **FAILOVER** соответствует системному объекту **Сервис отказоустойчивости**.

От объекта **FAILOVER** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	Описание событий
START	Объекты перенесены на резервный Сервер
STOP	Объекты возвращены на основной Сервер

Список команд и параметров для объекта **FAILOVER** представлен в таблице:

Команда – описание команды	**Комментарий**

"FORCED_START" – принудительный перенос конфигурации основного Сервера на резервный	Обратный перенос конфигурации выполняется по команде FORCED_STOP или при перезагрузке/переподключении основного Сервера
"FORCED_STOP" – принудительный перенос конфигурации с резервного Сервера на основной	

7.28 OPERATORPROTOCOL Протокол оператора

Объект **OPERATORPROTOCOL** соответствует системному объекту **Протокол оператора**.

От объекта **OPERATORPROTOCOL** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события	Параметры события
ACTIVATE_LEFT	Оператор кликнул левой кнопкой мыши по ячейке события в окне **Протокола оператора**	
ACTIVATE_RIGHT	Оператор кликнул правой кнопкой мыши по ячейке события в окне **Протокола оператора**	
POSTPONE_PRESSED	Оператор нажал на кнопку **Отложить**	
CREATE_REPORT	Оператор нажал на кнопку **Сформировать** на вкладке **Создать отчет**	В параметре user_id<> указан идентификатор пользователя. В параметрах initial_date<> и final_date<> указаны выбранные в интерфейсе начальная и конечная даты
RESPONSE_ALARM	Оператор нажал на кнопку **Тревожная ситуация**	objtype<> – Тип объекта objid<> – Идентификатор объекта action<> – Название события в Базе данных alarm_time<> – Время возникновения тревоги
RESPONSE_SUSPECT	Оператор нажал на кнопку **Подозрительная ситуация**	
RESPONSE_FALSE	Оператор нажал на кнопку **Ложное срабатывание**	
ACTIVATE_EVENT	Фокусировка на событии: клик по событию в интерфейсе или переход к нужному событию с помощью клавиатуры	

Список команд и параметров для объекта **OPERATORPROTOCOL** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"DEL_ALARM" – удалить тревогу	objtype<>	Тип объекта (например, CAM, GRELE и т.д.)
	objid<>	Идентификатор объекта
	options<>	Возможные значения: • first – удалить первую тревогу • last – удалить последнюю тревогу • all либо пусто – удалить все тревоги
"HIDE_BUTTON" – скрыть кнопки присвоения статуса событию	button<>	Названия кнопок через запятую: • alarm – Тревожная ситуация • suspicious – Подозрительная ситуация • false – Ложное срабатывание Пример задания параметра:

		button<alarm,suspicious,false>
	hide<>	1 – скрыть кнопки, перечисленные в параметра button 0 – отобразить кнопки, перечисленные в параметра button
	objtype<>	Тип объекта
	objaction<>	Тип события
	objid<>	Идентификатор объекта

7.29 PERSON Пользователь

Объект **PERSON** соответствует системному объекту **Пользователь**.

От объекта **PERSON** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

Событие	**Описание события**
REGISTERED	Вход пользователя в систему
UNREGISTERED	Выход пользователя из системы

7.30 IPJOYSTICK Устройство управления

Объект **IPJOYSTICK** соответствует объекту **Устройство управления**.

От объекта **IPJOYSTICK** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

События	**Параметр**	**Описание параметра**
KEY_PRESSED – Нажата клавиша	button<>	Код клавиши

7.31 CAM_FACECAPTURE Детектор лиц

Объект **CAM_FACECAPTURE** соответствует системному объекту **Детектор лиц**.

От объекта **CAM_FACECAPTURE** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

События	**Описание событий**
FACE_DETECTED	Лицо захвачено
FACE_LEAVE	Лицо потеряно

Список параметров для объекта **CAM_FACECAPTURE** представлен в таблице:

Параметры	Описание параметров
owner	Имя сервера, на котором произошел захват\потеря лица
fraction	Миллисекунда захвата/потери лица

Параметры	Описание параметров
module	Модуль, на котором происходит захват
date	Дата захвата/потери лица
guid_pk	ID события (генерируется случайным для каждого события)
core_global	Раздать всем ядрам (оповестить всех)
guid	ID захваченного\потерянного лица (генерируется случайным для каждого события)
time	Время захвата/потери
param0	Аналогично параметру **guid**. Используется для вывода информации в столбце «Доп. Инфо» протокола событий

7.32 EVENT_VIEWER Протокол событий

Объект **EVENT_VIEWER** соответствует системному объекту **Протокол событий**.

От объекта **EVENT_VIEWER** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	**Описание события**
SHOW_ON_MAP	Оператор выбрал команду «Показать на карте»
SHOW_VIDEO	Оператор выбрал команду «Показать видео»
SHOW_REPORT	Оператор выбрал команду «Вывести отчет»
CREATE_REPORT	Генерируется, если ключ реестра GenerateEventInsteadOfReport установлен равным 1 и оператор выбрал команду «Вывести отчет». При этом сам отчет не открывается. См. также Справочник ключей реестра

Список команд и параметров для объекта **EVENT_VIEWER** представлен в таблице:

Команда – описание команды	**Параметры**	**Описание параметров**
"UPDATE_VIEW" – установить общий цвет фона и/или текста в окне Протокола событий	bk_color<>	Общий цвет фона в 16-ричном виде
	defclr<>	Общий цвет текста 16-ричном виде

7.33 CAM_TITLE Титрователь

Объект **CAM_TITLE** соответствует системному объекту **Титрователь**.

Список команд и параметров для объекта **CAM_TITLE** представлен в таблице:

Команда – описание команды	**Комментарий**
"REINDEX" – запустить обновление базы банных титров	При любом значении параметра _id_ в команде DoReact() запускается переиндексация базы данных титров. См. также Утилита конвертации базы данных титров Cam_title_updater.exe

7.34 IPSTORAGE Внешнее хранилище

Объект **IPSTORAGE** соответствует системному объекту **Внешнее хранилище**.

Список команд и параметров для объекта **IPSTORAGE** представлен в таблице:

Команда – описание команды	**Параметр**	**Описание параметра**	**Комментарий**
IMPORT – импорт недостающей части архива за заданный период	cam<>	Идентификатор камеры	Команда применяется в случае, если по каким-то причинам не был выполнен автоматический импорт из внешнего хранилища. *Примечание. Если в момент отправки реакции выполняется импорт, то команда не сработает. Чтобы прервать текущую задачу импорта, необходимо сначала отправить команду UPDATE_TIME*
	datetime_from<>	Дата и время, начиная с которых следует выполнять импорт, в формате <ДД-ММ-ГГ ЧЧ:ММ:СС>	
	datetime_to<>	Дата и время, до которых следует выполнять импорт, в формате <ДД-ММ-ГГ ЧЧ:ММ:СС>	
UPDATE_TIME – остановить синхронизацию и задать время последнего импорта из внешнего хранилища в файле Settings.xml	cam<>	Идентификатор камеры	Команда применяется, когда необходимо остановить текущую задачу импорта и выполнить команду IMPORT для синхронизации заданного периода архива
	datetime<>	Дата и время последней синхронизации, которые необходимо установить в файле Settings.xml	

7.35 TELEGRAM Telegram бот

Объект **TELEGRAM** соответствует системному объекту **Telegram бот**.

От объекта **TELEGRAM** поступают события, представленные в таблице. Запуск процедуры происходит при возникновении соответствующего события.

Событие	**Описание события**	**Комментарий**
ERROR	Ошибка отправки сообщения	В параметре error<> содержится текстовое описание ошибки

Список команд и параметров для объекта **TELEGRAM** представлен в таблице:

Команда – описание команды	**Параметры**	Описание параметров
"SEND" – отослать	text<>	Текст сообщения
	chat_id<>	Идентификатор чата
	bot_id<>	Идентификатор бота
	longtitude<>	Долгота геолокации
	latitude<>	Широта геолокации
	address<>	Текстовый адрес геолокации
"SENDPHOTO" – отослать фото	photo<>	Полный путь к файлу изображения
	caption<>	Подпись к файлу
	chat_id<>	Идентификатор чата
	bot_id<>	Идентификатор бота
	longtitude<>	Долгота геолокации

Команда – описание команды	Параметры	Описание параметров
	latitude<>	Широта геолокации
	address<>	Текстовый адрес геолокации

7.36 BACNET BacNet

Объект **BACNET** соответствует системному объекту **BacNet**.

От объекта **BACNET** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание события
ERROR	Получено сообщение об ошибке
EVENT_OCCURES	Подтверждение получения сообщения
WRITE_OCCURES	Подтверждение выполнения записи
WRITE_RESULT	Результат выполнения записи

Список команд и параметров для объекта **BACNET** представлен в таблице:

Команда – описание команды	Параметры	Описание параметров
"WRITE" – отправить значение в устройство BacNet	bacnet_application_tag<>	Тип данных. Возможные значения: NULL = 0 BOOLEAN = 1 UNSIGNED INT = 2 SIGNED INT = 3 REAL = 4 DOUBLE = 5 OCTET STRING = 6 CHARACTER STRING = 7 BIT STRING = 8
	bacnet_value<>	Значение параметра
	bacnet_objtype<>	Тип объекта: ANALOG INPUT = 0 ANALOG OUTPUT = 1 ANALOG VALUE = 2 BINARY INPUT = 3 BINARY OUTPUT = 4 BINARY VALUE = 5
	bacnet_instance<>	Уникальный глобальный идентификатор устройства BACnet
	bacnet_property_id<>	Идентификатор свойства
	bacnet_device_id<>	Идентификатор устройства BACnet в системе

"EVENT" – отправить сообщение в устройство BACnet	event_type<>	Тип события
	from_state<>	Перевод из состояния
	to_state<>	Перевод в состояние
	message_text<>	Название события

7.37 CAM_IP_DETECTOR Детектор встроенный

Объект **CAM_IP_DETECTOR** соответствует системному объекту **Детектор встроенный**.

От объекта **CAM_IP_DETECTOR** поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

Описание событий от объекта **CAM_IP_DETECTOR**:

Событие	Описание событий	Комментарий
Detected	Событие	Событие поступает при получении метаданных от встроенных детекторов. Например, данное событие отображается при получении данных о температуре тела от тепловизора, и т.п. Также событие поступает, если данные от детекторов передаются с использованием Onvif. В параметре param0<> содержится строковое значение, содержащее параметры события. Передаваемые параметры зависят от встроенного детектора; если используется Onvif, то содержимое параметра можно настроить на вкладке **Настройки событий ONVIF-Сервера** (см. Фильтрация событий ONVIF-Сервера).

Примеры событий от встроенных детекторов:

Пример 1

```
// Событие от тепловизора
Event : CAM_IP_DETECTOR|1|DETECTED|slave_id<QA-T51>,
fraction<16>,owner<QA-T51>,module<video.run>,date<23-04-20>,
guid_pk<{1345DC60-3485-EA11-8A95-B06EBF8119EF}>,core_global<1>,time<10:31:06>,
param0<TargetList:name=TargetList;type=6;TemperatureValue0:37.4;json0:{
   "BeginTime" : "20200423T073058.000000",
   "EndTime" : "20200423T073100.000000",
   "EventClass" : "FaceEvent",
   "Hypotheses" : [
      {
         "Age" : 0,
         "BestTime" : "20200423T073059.000000",
         "Gender" : "unknown",
         "Rectangle" : [ 0.6380, 0.550, 0.0680, 0.1560 ],
         "TemperatureValue" : 37.40
      }
   ],
   "Id" : 1
}
;>
```

Пример 2

```
// Событие от детектора VMDA
Event:CAM_IP_DETECTOR|1|DETECTED|param0<Comment:ver_type<0>,objtype<SLAVE>,int_obj_id<1>,module
<video.run>,
core_global<1>,_TRANSPORT_ID<>,time<12:22:30>,objaction<PING>,onvif_event<>,
date<30-03-21>,slave_id<DESKTOP-JHRURJJ>,
objid<DESKTOP-JHRURJJ>;>,int_obj_id<1>,core_global<1>,
guid_pk<{9A989C70-3991-EB11-BDFF-00155DF96D00}>,slave_id<DESKTOP-339SH3U>,time<12:22:30>,_times
tamp<7520749>,
fraction<465>,date<30-03-21>,owner<DESKTOP-339SH3U>,module<video.run>
```

7.38 SIP_TERMINAL SIP-терминал

Объект **SIP_TERMINAL** соответствует системному объекту **SIP-терминал**.

От объекта **SIP_TERMINAL** поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание	Содержание параметра param0<>, отображаемого в поле **Доп. инфо** в Протоколе событий
CALL_END	Конец вызова	Номер абонента, который звонил
CALL_END_OPERATOR	Конец связи с оператором	Номера абонентов и продолжительность вызова. Например, если параметр принимает значение "903 to 906 (01:04)", это означает, что абонент 903 звонил абоненту 906, и звонок длился 1 минуту и 4 секунды
CALL_END_DEVICE	Конец связи с устройством	
CALL_END_VIRTUAL	Конец связи с особым номером	
CALL_BEGIN	Начало вызова	Номера абонентов: совершающий вызов и тот, кому звонят
CALL_TRYING	Попытка вызова	
CALL_BEGIN_VIRTUAL	Начало вызова особого номера	
CALL_TRYING_VIRTUAL	Попытка вызова особого номера	

Список команд и параметров для объекта SIP_TERMINAL представлен в таблице.

Команда – описание команды	**Параметры**	**Описание параметров**
"END_ALL_CALLS" – завершить все звонки на указанном терминале (независимо от того, установлено ли соединение)	-	-

7.39 INC_MANAGER Менеджер инцидентов

Объект INC_MANAGER соответствует системному объекту **Менеджер инцидентов**.

От объекта INC_MANAGER поступают события, представленные в таблице. Запуск процедур происходит при возникновении соответствующего события.

События	**Описание событий**	**Параметры**	**Описание параметров**	**Комментарий**
CLOSE_CLICK	Нажатие кнопки **Закрыть** в интерфейсе			Событие генерируется, когда в интерфейсе **Менеджера инцидентов** инцидент

События	Описание событий	Параметры	Описание параметров	Комментарий
				закрывается без обработки оператором
CLOSE_ALL_CLICK	Нажатие кнопки **Закрыть все** в интерфейсе			Событие генерируется, когда в интерфейсе **Менеджера инцидентов** все инциденты закрываются без обработки оператором
SELECT	Клик по инциденту в интерфейсе			Событие генерируется, если в интерфейсе **Менеджера инцидентов** оператор кликнул по событию правой или левой кнопкой мыши
ACTIVATE_EVENT	Событие выделено Оператором (клик по событию мышью)	alarm_time<>	Время возникновения события	
		event_guid<>	Идентификатор события (генерируется случайным для каждого события)	
		objtype<>	Тип объекта (например, CAM, GRELE и т.д.)	
		action<>	Тип действия (например, MD_START, DISARM и т.д.)	

7.40 INC_SERVER Сервер инцидентов

Объект INC_SERVER соответствует системному объекту **Сервер инцидентов**.

От объекта INC_SERVER поступают события, представленные в таблице ниже. Запуск процедур происходит при возникновении соответствующего события.

Событие	Описание событий	Комментарий
EVENT	Событие (инцидент) взято в обработку в **Менеджере инцидентов** или идёт обработка оператором	Событие генерируется: 1) когда оператор берёт событие в обработку; 2) на каждом шаге обработки события. Параметр **serializeBase64** события содержит JSON с подробной информацией об обрабатываемом событии, в том числе сделанные шаги оператора

Список команд и параметров для объекта INC_SERVER представлен в таблице:

Команда – описание команды	Параметры	Описание параметров	Комментарий
"UPDATE_STATUS" – изменить статус события (инцидента) в **Менеджере инцидентов**	pks<>	Массив идентификаторов событий	Параметры являются фильтрами и наличие хотя бы одного параметра обязательно. Значения параметра можно указать через разделитель. Это означает, что будет выбрано одно ИЛИ другое значение. Пример: objids<1|2>
	objtypes<>	Типы объектов	
	objids<>	Идентификаторы объектов	
	actions<>	Действия	

Команда – описание команды	**Параметры**	**Описание параметров**	**Комментарий**
	status<>	Статус события: 0 – Ожидает обработки 1 – В работе 2 – Приостановлен 3 – Завершено	
"UPDATE_ESCALATE_STATUS" – изменить статус эскалации события (инцидента) в **Менеджере инцидентов**	escalated<>	Статус эскалации события: 0 - Ожидает обработки (не эскалировано) 1 - Эскалировано	
	pks<>, objtypes<>, objids<>, actions<> те же, что и для UPDATE_STATUS		

8 Заключение

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

1. Руководство администратора: настройка системных объектов в интерфейсе.
2. Руководство оператора: работа с ПК *Интеллект*.
3. Руководство по установке и настройке компонентов охранной системы: установка и настройка периферии (камеры, сигнализация, системы контроля доступа и т.д.).

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?

Чем более полную и подробную информацию вы предоставите, тем быстрее наши специалисты смогут устранить 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.