Guia de Programación

Guía de programación. Introducción

Objeto del software INTELLECT™

El software INTELLECT™ ha sido diseñado para su utilización en sistemas de seguridad industriales integrados, flexibles (ajustables) y escalables que estén basados en sistemas de videovigilancia digital y monitoreo de audio.

El software INTELLECT™ cuenta con las siguientes funciones básicas:

1. Integración de sistemas de videovigilancia digital y monitoreo de audio con los sistemas de datos existentes, varios equipos de seguridad y software de terceros, utilizando interfaces abiertas integradas en el intercambio de datos.
2. Compatibilidad con diversos dispositivos de seguridad y sistemas de datos, particularmente con la alarma de incendio y seguridad y los sistemas de control de accesos, cámaras de videovigilancia, sistemas de análisis de datos y sistemas para el reconocimiento de objetos (eventos) y su identificación mediante imagen.
3. Registro de fuente única y procesamiento de eventos, generación de notificaciones y control de respuestas de acuerdo con la lógica modificada de forma flexible.
4. Capacidades prácticamente ilimitadas para el escalamiento, ajustes para soluciones concretas, redistribución de recursos con cambios en la cantidad o calidad de las tareas al supervisar lugares vigilados y operar varios equipos.

Configuración de las interacciones lógicas entre objetos en Intellect

Las capacidades de *Intellect* se basan en interacciones lógicas entre objetos. La siguiente tabla contiene información general sobre las diferentes formas de configurar las interacciones lógicas:

Método de configuración de la interacción lógica	**Descripción**	**Implementación**	**Ejemplo**
Ajustes de los paneles de objetos de sistema	Configuración base de la interacción entre objeto del sistema	Implementación por medio de la funcionalidad de los objetos del sistema – ver Guía del Administrador	Configuración de la imagen en pantalla en el **Monitor**
Macro	Configuración de interacciones simples entre objetos si su funcionalidad no permite realizar las acciones necesarias	Implementación por medio del objeto **Macro** – ver Guía del Administrador	Habilitar el actuador (relé) cuando el sensor está cerrado
Programa	Configuración de interacciones complejas entre objetos si la funcionalidad del objeto **Macro** no permite realizar las acciones necesarias	Implementación por medio del objeto **Programa** como código en el lenguaje de programación integrado – ver esta guía	Restablecer las cámaras PTZ y realizar captura cada 15 minutos
Script		Implementación por medio del objeto **Script**como código JScript – ver Guía de Programación (JScript)	

Objetivo y estructura de la guía

La Guía de Programación es un manual de referencia e información sobre cómo programar en el lenguaje integrado del software Intellect. Esta guía ha sido diseñada para administradores de sistemas, técnicos de instalación y configuración, usuarios con derechos de administrador en sistemas de videovigilancia digital y monitoreo de audio, y desarrollada a partir del software INTELLECT™.

La programación en INTELLECT™ habilita el control automático del sistema al configurar las interacciones lógicas complejas entre objetos.

La guía recoge información acerca de:

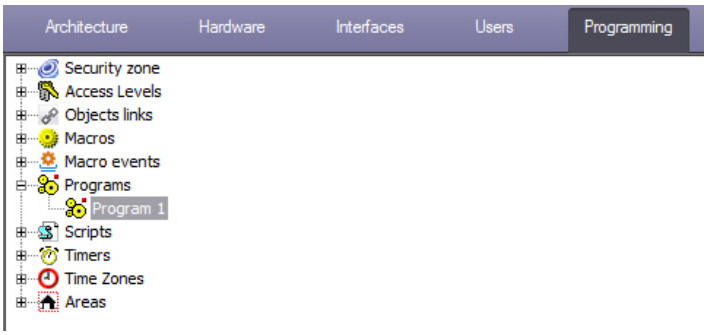
1. Herramientas de programación;
2. Descripción de la sintaxis del lenguaje de programación integrado;
3. Ejemplos de programas en el lenguaje integrado.

Herramientas de programación en Intellect

El objeto de sistema Programa

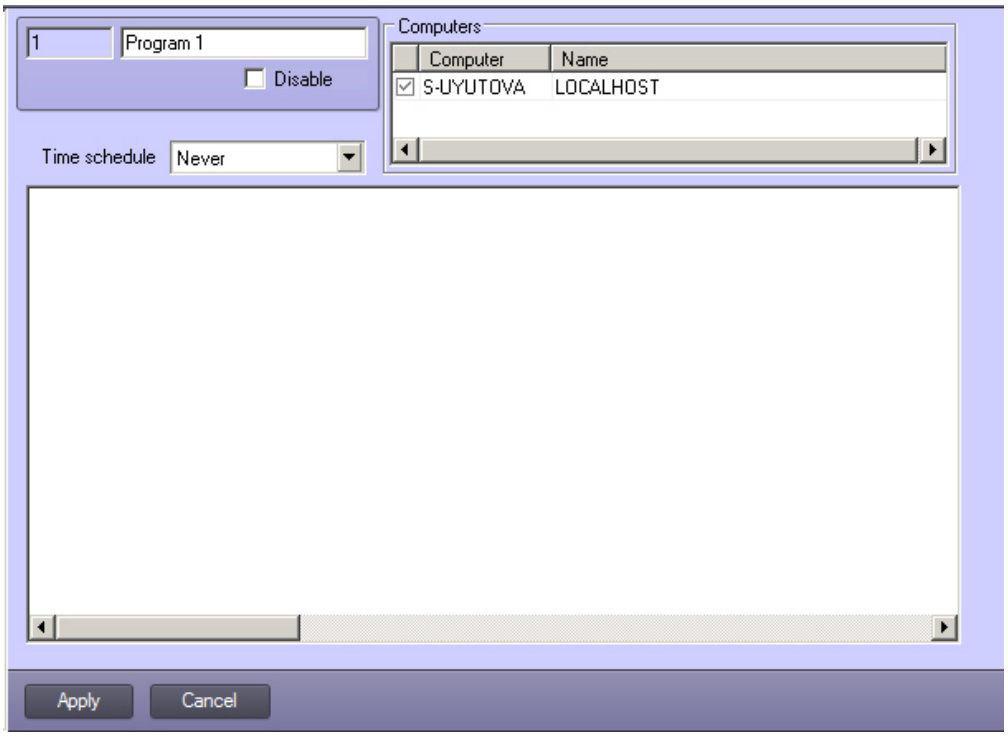
El objeto de sistema **Programa** está diseñado para iniciar el programa escrito en Jscript (o lenguaje de programación Intellect) en Intellect y establecer sus parámetros.

El objeto de sistema **Programa** se crea bajo el objeto **Programas** en la pestaña de **Programación** del cuadro de diálogo **Configuración del sistema**.



¡Importante!
La creación de más de 100 objetos de sistema **Programa** puede causar inestabilidad en el sistema.

El cuadro de abajo muestra el panel de configuración del objeto de sistema **Programa**:



Especifique la zona horaria de ejecución del programa y los equipos (núcleos) en los que se va a ejecutar en el panel de configuración del objeto de sistema **Programa**.

Nota.
Para activar todas las casillas de verificación, elija una en la columna y presione Ctrl+A. Para desactivar todas las casillas de verificación, elija una y presione Shift+A.

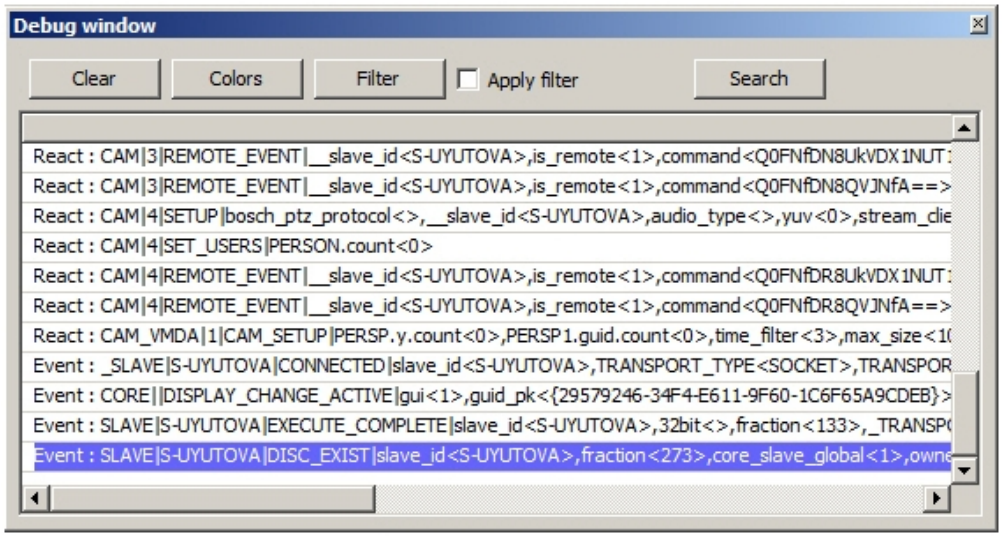
El procesador de textos se encuentra en el panel de configuración del objeto de sistema **Programa**. Esta herramienta se usa para escribir y editar el código del programa.

Se puede deshacer o rehacer algunas acciones utilizando combinaciones en el procesador de textos en el panel de configuración del objeto de sistema **Programa**. Para deshacer alguna acción presione **Alt**+**Backspace** (Retroceso), para rehacer – **Ctrl**+**Y**.

La ventana de Depuración (Debug)

La ventana de Depuración (Debug) está diseñada para visualizar datos acerca de todos los eventos registrados en el sistema.

Para abrir la ventana de**Depuración**, utilice el comando **Depuración** en el menú **Ejecutar** en el Panel de Control Principal. La ventana de **Depuración** de Intellect aparece en la parte inferior de la pantalla.



Por defecto, la **ventana de Depuración** no está disponible. Para habilitar la **ventana de Depuración**, use la utilidad *twea ki.exe* (ver la sección La ventana de Depuración de la Guía de Programación (JScript)).

Analizador sintáctico

El analizador sintáctico integrado permite comprobar la ortografía de palabras básicas que hayan sido registradas como, por ejemplo, OnEvent, DoReact, OnTime, Wait, Sleep, etc. Se debe tener en cuenta que el analizador no comprueba si los parámetros de comando están escritos de forma correcta, por lo que se deberá prestar mucha atención en estos casos.

```
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+">");
```

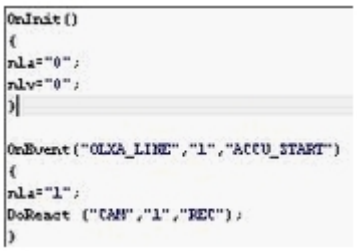
Para cambiar el tamaño de la fuente, utilice las combinaciones

CTRL y **+** para aumentar el tamaño de la fuente

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

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

CTRL y **–** para reducir el tamaño de la fuente



Procedimiento recomendado para escribir programas

En la página:

1. Establecer una tarea general
2. Estructurar la tarea en subtareas
3. Escribir subtareas y depúrelas
4. Encontrar y corregir errores

Establecer una tarea general

Es necesario que exista una visión clara de lo que se debe hacer en el sistema como resultado de eventos específicos. Especifique la ID de los dispositivos que participan la creación de eventos y acciones.

Estructurar la tarea en subtareas

Si se deben procesar varios eventos en una única tarea, aquello que se debe hacer con cada evento debe estar claro. Siempre que sea posible, elimine la posibilidad de ejecución de secuencia de comandos en bucle, es decir, excluya cualquier acción recursiva si no está relacionada con la ejecución de la tarea.

Escribir subtareas y depurarlas

La parte más compleja de escribir scripts es crear la lista de acciones con un posible uso de operaciones lógicas y de bucle. La depuración de esta parte de la programación lleva mucho tiempo. La generación de eventos que necesita procesamiento no suele ser sencilla de utilizar, especialmente con un objeto real como, por ejemplo, hacer saltar un sensor de incendios – desde el servidor al núcleo del sistema. En este caso, se recomienda generar un evento manualmente en la fase de depuración y la mejor forma es ejecutar una macro vacía. Tras depurar el cuerpo de la secuencia de comandos, lo que queda es un evento real en lugar de ejecutar la macro vacía. Además, se puede comprobar y viceversa – asegúrese de que el evento está escrito correctamente sin iniciar la lista de acciones – ejecute una macro vacía y observe su funcionamiento en la ventana de depuración.

Encontrar y corregir errores

Al iniciar, el analizador sintáctico incorporado comprueba si los nombres de las funciones estás escritos correctamente, pero no comprueba la posición de los caracteres clave – comas, signos de punto y coma, paréntesis anidados, etc. Por tanto, los errores, si los hay, serán evidentes sólo en la fase de ejecución del cuerpo del programa.

Descripción de sintáxis

El script consta delconjunto de procedimientos.

Todos los operadores ejecutados en los procedimientos están incluidos en {..} bloques.

Si se tiene que escribir un comentario, se debe incluir // caracteres reservados antes del comentario.

Descripción de las variables

Todas las variables del sistema son variables de secuencia.

Para comparar las variables de secuencia y los valores, utilice la función boolstrequal (string1, string2). La función strequal devuelve un valor distinto a cero si las secuencias son iguales (ver la sección Descripción de las funciones).

Utilice la función str (string1) para realizar operaciones completas (ver la sección Descripción de las funciones).

Descripción de los procedimientos

Procedimientos estándar

Existen 3 procedimientosestándar que se puedenrealizar cuando ocurra el evento correspondiente:

1. Onlnit() – se usa para la inicialización de las variables (configurar los valores iniciales) que se utilizarán en los sripts. Se debe ejecutar antes de iniciar cualquier módulo del sistema. Se recomienda nombrar el procedimiento una única vez para todos los scripts. Ejemplo de uso:

```
OnInit(){
   flag=1;
   num=8;  //se inicializarán las variables en el inicio
}
```

2. OnTime (DOW (1-7), día-mes-año, horas, minutos, segundos) – Funcionamiento en un momento específico.

```
OnTime(W,D,X,Y,H,C,S)
{
//W - DOW (0 – Lunes, 6 - Domingo);
//D – fecha en el formato día-mes-año, 16 Agosto 2001 es "16-08-01"
//X,Y - reservado

            //H - hora
            //C - minutos
            //S - segundos
            // COMPARANDO CON LOS PARÁMETROS, LA ACCIÓN ESTÁ ESPECIFICADA EN
ADELANTE
}
```

Ejemplos de uso:

```
OnTime(W,"16-08-01",X,Y,"11","11","30")
{
   //el código se ejecutará el 16 Agosto, 2001 at 11:11:30
}
```

```
OnTime(W,D,X,Y,"11","11","30")
{
   //el código se ejecutará cada día a las 11:11:30
}
```

```
OnTime(W,"16-08-01",X,Y,H,C,S)
{
  //el código se ejecutará el 16 Agosto, 2001
  //cada segundo
}
```

```
OnTime(W,"16-08-01",X,Y,"11","11",S)
{
  //el código se ejecutará el 16 Agosto, 2001
  //cada segundo de 11:11 a 11:12
}
```

```
OnTime("0",D,X,Y,"21","0","0")
{
  //el código se ejecutará cada Lunes
  // a las 21:00:00
}
```

3. OnEvent (tipo de fuente, número, evento) – funciona si existe un evento específico proveniente del objeto de sistema. Éste es el procedimiento principal de creación de scripts. Ejemplos de uso:

```
OnEvent("GRAY","1","ON")
{
   //se ejecutará al cerrar el sensor 1
}
```

```
OnEvent("CAM","12","MD_START")
{
   //se ejecutará cuando la herramienta de detección de movimiento de la cámara 12 se dispare
}
```

Se puede ver cada procedimiento que posea parámetros en un código numerosas veces con varios parámetros. Cuando un evento ocurra, el sistema ejecutará aquellos que tengan los mismos parámetros que presente lo que ha ocurrido.

Se puede definir o no el parámetro del procedimiento. Si se define, el valor debe aparecer entre comillas, de lo contrario el parámetro se escribirá usando el alfabeto latino y el procedimiento se ejecutará en todoslos eventos para los que ha sido definido.

Ejemplos de uso:

```
OnEvent("GRAY","1","ON")       // se ejecutará al cerrar el sensor 1
{
   i=1;
   i=i+1;     //como las variables son de cadena, la suma será 11
   j=1;
   j=str(j+1); // str es una función de conversión de número a cadena. Dentro destr
            //función todas las variables de cadena (si las hay) se convierten en enteros y
            //todos los enteros se añaden juntos, por tanto, la suma será 2.
}
```

```
OnEvent("GRAY",N,"ON")  // se ejecutará cuando cualquier sensor esté cerrado
{
   if(strequal(N,"3")
   {
     // se ejecutará si éste es el sensor 3
   }
}
```

Creación de procedimientos personalizados

Todos los procedimientos personalizados descritos en el script deben aparecer en el mismo cuerpo del programa y antes de los procedimientos en los cuales se han nombrado.

```
procedure NombreProcedimiento(lista de parámetros){
   //cuerpo del procedimiento
}
```

¡Importante!
Los nombres de los parámetros deben constar de una letra en mayúscula.

Ejemplos de uso:

```
procedure NombreProcedimiento(A,B)
{
   n=A+" "+B;
   //cuando esté en funcionamiento macro 1 n=«Macro 1», cuando esté en funcionamiento macro
16 n=«Macro 16»
}

OnEvent("MACRO",N,"RUN")
{
   a1=N;
   a2="Macro";
   NombreProcedimiento(a2,a1);
}
```

Descripción de los operadores

La lista de operadores utilizados para describir acciones:

1. DoReact (tipo de objeto, número, acción[,Parámetros]) – ejecutar acción
   Ejemplo de uso:

```
OnEvent("GRAY","1","ON")
{
  DoReact("GRELE","1","ON");    //cerrar el relé 1 al cerrar el sensor 1
}
```

2. DoCommand(línea de comando) – ejecutar la línea de comando
   Ejemplo de uso:

```
OnEvent("GRAY","1","ON")
{
   DoCommand("notepad.exe"); //cuando el sensor 1 está cerrado ejecutar "Notepad"
}
```

3. Wait(número de segundos) – Esperar N segundos;
   Sleep(número de milisegundos) – Esperar N milisegundos.
   Los operadores de espera deben aparecer en un hilo único. El hilo único debe quedar en el interior de corchetes.
   Ejemplo. Cuando el Sensor 1 está cerrado, el Relé 1 se cierra 5 segundos.

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

4. Comprobar la función de estado:
   CheckState (tipo de objeto, número, estado) – si el estado de un objeto es objetivamente preciso, el resultado es 1; de otro modo, será 0.
   Se pueden usar expresiones como parámetros. Los valores constantes se deben entrecomillar.
   Ejemplo. Comprobar el estado de la cámara 2 al cerrar el sensor 1 y, si el estado es "Alarmed", cerrar el relé 1

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

5. Operador condicional

```
If(expresión)
{
   … // si el resultado no es igual a 0
}
else
{
   … // si el resultado es igual a 0
}
```

   Se puede omitir la parte else{}

   Ejemplo de uso:

```
OnEvent("MACRO","1","RUN")
{
  x=5;
  si(x>10) {y=2;}   // si "x" es superior a 10, entonces y=2
  o bien{y=3;}      //de otro modo y=3
}
```

6. Operador For:

```
For(expression 1; expression 2; expression 3){
            …
}
```

   La expresión 1 se ejecuta al principio del bucle; el cuerpo del bucle se ejecuta si la expresión 2 es verdadera; la expresión 3 se ejecuta después de cada ejecución del cuerpo del bucle.

Ejemplo. Cuando el sensor 1 está cerrado, el relé 1 se cierra y se abre cada segundo y esto pasará 10 veces.

```
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 (tipo de objeto, número, estado) – función que genera reacciones de los objetos del sistema. Mientras tanto, la reacción generada se envía a todos los núcleos conectados a través de la red.

   Ejemplo. Cuando se ejecuta la macro 1, la cámara se arma.

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

8. NotifyEventGlobal (tipo de objeto, número, estado) – función que genera eventos. Mientras tanto, los eventos generados se envían a todos los núcleos conectados a la red.
   Ejemplo. Al ejecutar la macro 1, se crea el evento "Grabación" en la cámara 1. El comando se envía a todos los núcleos como evento para su registro

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

Nota
Si no es necesario enviar un evento a todos los núcleos, utilice la función NotifyEvent.

Operadores y expresiones

La siguiente tabla establece una lista y descripción de los operadores de comparación, aritméticos y condicionales.

Operador	Descripción general, ejemplo de uso
Operadores de comparación	
>	Operador de comparación – mayor que. Ver ejemplo en la sección Descripción de los operadores.
<	Operador de comparación – menor que. Ver ejemplo en la sección Descripción de los operadores.
Operadores aritméticos	

+	Adición. Ejemplo de uso:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`x=5;`
	`y=10;`
	`i=x+y; // añade cadenas, i.e. 5+10=510`
	`e=str(x+y); // añade enteros 5+10=15`
	`}`

-	Sustracción. Ejemplo de uso:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`x=5;`
	`y=10;`
	`i=x-y; // sustrae enteros 5-10=-5`
	`e=str(x-y); // sustrae enteros 5-10=-5`
	`}`

*	Multiplicación. Ejemplo de uso:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`x=5;`
	`y=10;`
	`i=x*y; // multiplica enteros 5*10=50`
	`e=str(x*y); // multiplica enteros 5*10=50`
	`}`

/	División. Ejemplo de uso:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`x=5;`
	`y=10;`
	`i=x/y; // divide enteros 5/10=0.5`
	`e=str(x/y); // divide enteros 5/10=0.5`
	`}`

%	Resto tras la división de enteros. Ejemplos de uso.
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`a=1120.0;`
	`b=100;`
	`e=a%b; // resto tras división entre enteros, i.e 1100 se divide entre 100 y 20 es el resto.`
	`// si la división no tiene resto, entonces el resultado es 0`
	`}`

( )	Conjunto de operadores aritméticos. Ejemplo de uso.
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`x=100/((5*8)/1.028);`
	`}`

Operadores lógicos	

&&	Operador lógico AND. Ejemplo de uso:
	`OnEvent ("MACRO","1","RUN")`
	`{`
	`a=1;`
	`b=2;`
	`z=3;`
	`if((a<b)&&(b<z))`
	`{`
	`y=1; //si es falso, entonceselse`
	`}`
	`else`
	`{`
	`x=0;`
	`}`
	`}`

!	Operador lógico de inversión. Ejemplo de uso:
	`OnEvent ("CAM",N,"MD_START")`
	`{`
	`if(!(strequal(N,”1”,)))`
	`{`
	`DoReact(“GRELE”,”1”,””ON)`
	`}`
	`else`
	`{`
	`DoReact(“GRELE”,”2”,””ON)`
	`}`
	`}`

Descripción de las funciones

La siguiente tabla ofrece una descripción general y ejemplos de uso de funciones matemáticas, funciones de conversión, además de funciones de formato y funciones de cadena.

Funciones **(El número de parámetros ejecutables está especificado en corchetes)**	**Descripción general, ejemplo de uso**
MATEMÁTICAS	

sin[1]	Función trigonométrica: el seno de un ángulo. Formato: y=sin(x); donde y – valor de la función, x – argumento de la función (en radianes) Ejemplo: y=sin(1.6) Evento recibido: Event : CORE VAR_CHANGED nt_obj_id<1>,value<0.997495>,name<y>,time<15:26:41>,date<21-09-04>
cos[1]	Función trigonométrica: el coseno de un ángulo. Formato: y=cos(x); donde y – valor de la función, x – argumento de la función (en radianes) Ejemplo: y=cos(2.2) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.588501>,name<y>,time<16:00:45>,date<21-09-04>
tan[1]	Función trigonométrica, devuelve la tangente de un ángulo. Formato: y=tan(x); donde y – valor de la función, x – argumento de la función (en radianes) Ejemplo: y=tan(1) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.557408>,name<y>,time<16:43:45>,date<21-09-04>
asin[1]	Devuelve el arcoseno de la expresión numérica especificada. Formato: y=asin(x); donde y – valor de la función (en radianes), x- argumento Ejemplo: y=asin(0.5) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.523599>,name<y>,time<16:46:39>,date<21-09-04>
acos[1]	Devuelve el coseno del arco de la expresión numérica especificada. Formato: y=acos(x); donde y – valor de la función (en radianes), x – argumento Ejemplo: y=acos(0.55) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.988432>,name<y>,time<16:46:39>,date<21-09-04>
atan[1]	Devuelve la tangente del arco de la expresión numérica especificada. Formato: y=atan(x); donde y – valor de la función (en radianes), x – argumento Ejemplo: y=atan(1.2) Evento recibido: Event : Event : CORE VAR_CHANGED int_obj_id<1>,value<0.876058>,name<y>,time<17:07:09>,date<21-09-04>

sinh[1]	La función sinh devuelve el seno hiperbólico del valor del argumento. Formato: y=sinh(x); donde y –valor de la función, x – argumento de la función Ejemplo: y=sinh(0.8) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.888106>,name<y>,time<17:12:26>,date<21-09-04>
cosh[1]	La función cosh devuelve el coseno hiperbólico del valor del argumento. Formato: y=cosh(x); donde y –valor de la función, x – argumento de la función Ejemplo: y=cosh(0.35) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<0.336376>,name<y>,time<17:25:25>,date<21-09-04>
tanh[1]	Función trigonométrica de un ángulo. Formato: y=tanh(x); donde y –valor de la función, x – argumento de la función Ejemplo: y=tanh(0.35) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<1.419068>,name<y>,time<17:25:25>,date<21-09-04>
exp[1]	Devuelve el valor de la funcióne^x, donde x – expresión numérica especificada. Formato: y=exp(x); donde y – valor de la función, x – argumento Ejemplo: y=exp(1.65) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<5.20698>,name<y>,time<17:39:22>,date<21-09-04>
log[1]	Devuelve el logaritmo natural (base-e) de la expresión numérica especificada. Formato: y=log(x); donde y–valor de la función, x – argumento Ejemplo: y=log(0.65) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<-0.430783>,name<y>,time<17:43:22>,date<21-09-04>
log10[1]	Devuelve el logaritmo común (base-10) de la expresión numérica especificada. Formato: y=log10(x); donde y–valor de la función, x – argumento Ejemplo: y=log10(0.05) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<-1.30103>,name<y>,time<17:46:28>,date<21-09-04>

sqrt[1]	Devuelve la raíz cuadrada de la expresión numérica especificada. Formato: y=sqrt(x); donde y–valor de la función, x – argumento Ejemplo: y=sqrt(9) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<3>,name<y>,time<17:25:25>,date<21-09-04>
abs[1]	La función abs devuelve el valor absoluto del argumento. Formato: y=abs(x); donde y–valor de la función, x – argumento. Ejemplo: y= abs(-1) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<1>,name<y>,time<13:39:37>,date<22-09-04>
deg[1]	Función trigonométrica de un ángulo. Devuelve la medida de grado. Formato: y=deg(x); donde y – valor de la función en grados, x – valor del argumento en radianes. Ejemplo: y=deg(3.14) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<179.908748>,name<y>,time<13:13:51>,date<22-09-04>
rad[1]	Función trigonométrica de un ángulo. Formato: y=rad(x); donde y – valor de la función en radianes, x – valor del argumento en grados. Ejemplo: y=rad(180) Evento recibido: Event : CORE VAR_CHANGED value<3.141593>,name<y>,time<15:04:17>,date<17-03-08>
CONVERSIÓN	
floor[1]	Función de conversión a enteros (redondeando *a la baja)*. Formato: x= floor(y); donde x – valor de la función, y- fracción o entero Ejemplo: x= floor(5.55) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<5>,name<x>,time<20:51:48>,date<21-09-04>
ceil[1]	Función de conversión a enteros (redondeando *al alza)*. Formato: x= ceil (y); donde x – valor de la función, y – fracción o entero. Ejemplo: x= ceil(5.55) Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<6>,name<x>,time<20:51:48>,date<21-09-04>

str[1]	Función de conversión de entero a cadena.
	Formato: x=str(y); donde x–valor de la función, y – argumento
	Ejemplo:
	z=(9);
	a=str(z);
	b=sqrt(a);
	Eventos recibidos:
	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]	Función de conversión de cadena a entero. Formato: x=atof(y); donde x–valor de la función, y – argumento Ejemplo: x="0"; x=str(atof(x)+10); Evento recibido: 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]	Función de conversión de entero a cadena. Formato: x=val(y); donde x–valor de la función, y – argumento Ejemplo: x="10"; x=str(val(x)+2); Eventorecibido: 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]	Conversión de fracción a entero (sin la parte fraccional)
	Formato: x=int(y); donde x –valor de la función, y – argumento (fracción para la conversión)
	Ejemplo:
	y=(2.33);
	x=int(y);
	Eventorecibido:
	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>

long2time[1]	Se usa para convertir el número especificado de segundos en tiempo. Formato: x=long2time(y); donde x –valor de la función(tiempo), y –número en segundos Formato de la grabación inicial (argumento): <MM> Formato de la grabación final: <HH:MM:SS> Ejemplo: x=long2time(12345); Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<03:25:45>,name<x>,time<13:53:02>,date<20-09-04>.

time2long[1]	Convertir tiempo en un número de segundos
	Formato: x=time2 long(y); donde x –valor en segundos, y–tiempo en formato <horas>.<minutos>.
	Ejemplo:
	y=(0.15);
	x=time2long(y);
	Eventorecibido:
	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]	Convertir un número de días en una fecha. (número de días partiendo de d.C.)
	Formato: x= scalar2date (y); donde– x- valor(fecha), y – número de días.
	Ejemplo:
	y=(731500);
	x=scalar2date(y);
	Eventorecibido:
	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]	Convertir una fecha en un número de días. (El número de días se calcula partiendo de d.C.) Formato: x=scalar(y); donde x –valor numérico (en días), y –fecha. Formato de grabación: <DD.MM.YYYY> Ejemplo: x=scalar("19.10.2004") Evento recibido: 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>

convert_num[1]	Convertir un número en la cadena.
	
	Formato: x=convert_num(y); donde x –valor de la cadena del número, y –número convertible.
	
	Ejemplo:
	
	y=(24009921);
	
	x=convert_num(y);
	
	Eventorecibido:
	
	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<Twenty four million nine thousand nine hundred twenty-one >,name<x>,time<12:37:20>,date<23-09-04>

convert_cur[1]	Conversión de un número (suma de dinero) en la cadena y añadir rublos y kopeks.
	
	Formato: x=convert_cur(y); donde x –valor de la cadena de la suma de dinero, y–número (suma de dinero).
	
	Formato de grabación: <RR.KK>
	
	Example:
	
	y=(17999.98);
	
	x=convert_cur(y);
	
	Eventorecibido:
	
	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<Seventeen thousand nine hundred ninty-nine roubles ninty-eight copecks>,name<x>,time<12:49:30>,date<23-09-04>

FORMATEADO	

number_frm[2]	Formateo de un número
	
	Formato: x=number_frm(y,z); donde x – valor de la función, y– número inicial, z –cantidad de números que siguen al decimal.
	
	Ejemplo:
	
	y=(17999.09998);
	
	x=number_frm(y,3);
	
	Eventorecibido:
	
	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>

int_frm[2]	Formateo de un número
	
	Formato: x=int_frm(y,z); donde x – valor, y – operando, z –número de dígitos resultante.
	
	Ejemplo:
	
	y=(17999.99);
	
	x=int_frm(y,10);
	
	Eventorecibido:
	
	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]	Formateodel valor de la moneda (de '.' a '-'). Formato: x=currency_std(y); donde x –valor de la función con formato modificado, y– número (suma de dinero). Ejemplo: x=currency_std(3.62); Evento recibido: Event : CORE VAR_CHANGED int_obj_id<1>,value<3-62>,name<x>,time<13:40:01>,date<23-09-04>
IsVarExist[1]	La función que comprueba un parámetro especificado en el evento. Formato: y=IsVarExist("x"); donde y –valor, x – parámetro Si existe el parámetro, devuelve "1", de lo contrario, devolverá "0" Ejemplo: p=IsVarExist("param0") Evento recibido: 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]	La función que devuelve la ID del primer objeto que se encuentre por parámetro especificado. Id=GetObjectIdByParam ("x","y","z"); donde id – valor devuelto, x – tipo de objeto, y – parámetro, z – valor del parámetro. Ejemplo: Id=GetObjectIdByParam("CAM","color","0"); Eventorecibido: 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 (seevalue<2>), si la función devuelve un valor vacío (value<>), compruebe si ésta y sus parámetros están escritos correctamente.
CADENAS	

strequal[2]	Comparación de cadenas
	
	Formato: x= strequal(z,y); donde x – valor, z + y – cadenas en comparación.
	
	Ejemplo:
	
	z=str(1019);
	
	y=str(1019);
	
	x=strequal(z,y);
	
	Evento recibido:
	
	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>» (ver los ejemplos de arriba) – en el evento recibido, obtenemos «value<>» - si las cadenas en comparación difieren, o «value<1>» - silas cadenas en comparación son idénticas.

strsub[2]	Indica si existe una subcadena en la cadena.
	
	Formato: x=strsub(y,z); donde x –valor, y – cadena en la que se realiza la búsqueda, z – subcadena.
	
	Ejemplo 1:
	
	z=str(888123);
	
	y=str(123);
	
	x=strsub(z,y);
	
	Evento recibido:
	
	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>
	
	Ejemplo 2:
	
	z="67hb8vc56";
	
	y="vc";
	
	x=strsub(z,y);
	
	Evento recibido:
	
	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>" (ver el ejemplo 1) – Indexar la cadena de inicio. Partiendo de este índice, se detecta la primera ocurrencia de la subcadena en la cadena. Si el resultado de la búsqueda es negativo, la función devuelve el valor<>.

strempty[1]	Definir si la cadena está vacía. Formato: x=strempty(y); donde x – valor (1 si la cadena está vacía), y – cadena. Ejemplo: y=""; x=strempty(y); Eventorecibido: 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> * el valor de la función<>significa que la cadena no está vacía.

straleft[2]	Alineado a la izquierda
	
	Formato: x=straleft(y,z); donde x – cadena alineada, y – cadena, z – valor del alineado.
	
	Ejemplo:
	
	y=str(123456789);
	
	x=straleft(y,5);
	
	Eventorecibido:
	
	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>
	
	Nota. Si z es mayor que el número de símbolos en la cadena, la función añade espacios a la derecha de la cadena inicial hasta que su longitud alcanza z.

strmid[3]	Obtener subcadena
	
	Formato: x=strmid(y,z,w); donde x – valor de la cadena, y- cadena, z – posición de la cadena, w – longitud de la cadena.
	
	Ejemplo:
	
	z=(7);//posición
	
	w=(9);//longitud
	
	x=strmid("get substring (1 - string, 2 - position, 3 - length)",z,w);
	
	y=strmid("get substring (1 - string, 2 - position, 3 - length)",17,10);
	
	Eventorecibido:
	
	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<substring>,name<x>,time<14:18:08>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<1 – string>,name<y>,time<14:18:08>,date<24-09-04>

strletf[2]	Obtener el *lado* izquierdo de una *cadena*
	
	Formato: y=strleft(s,w); donde y –valor de la cadena, s- cadena, w- longitud (desde el inicio de la cadena)
	
	Ejemplo:
	
	w=(5);//longitud
	
	s=("Get left *side* of *string*");//cadena
	
	y=strleft(s,w);
	
	Eventorecibido:
	
	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< Get left *side* of *string*>,name<s>,time<14:54:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Get>,name<y>,time<14:54:31>,date<24-09-04>

strright[2]	Obtener el lado derecho de una cadena (1 –cadena, 2 –longitud)
	
	Formato: y=strleft(s,w); donde y –valor de la cadena, s – cadena, w – longitud(desde el final de la cadena)
	
	Ejemplo:
	
	w=(6);// longitud
	
	s=("Get right *side* of *string*");//
	
	y=strright(s,w);
	
	Eventorecibido:
	
	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< Get right *side* of *string*>,name<s>,time<15:10:36>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<strings>,name<y>,time<15:10:36>,date<24-09-04>

strnleft[2]	Obtener sin el *lado*izquierdo de la *cadena*.
	
	Formato: y=strnleft(s,w); donde y –valor de la cadena, s – cadena, w –longitud del lado izquierdo que se cortará.
	
	Ejemplo:
	
	w=(6);//longitud
	
	s=("get without left side of string");//cadena
	
	y=strnleft(s,w);
	
	Eventorecibido:
	
	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<get without left side of string>,name<s>,time<15:32:38>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<without left side of string>,name<y>,time<15:32:38>,date<24-09-04>

srtnright[2]	Obtener sin el *lado* derecho de la *cadena*.
	
	Formato: y=strnright(s,w); donde y –valor de la cadena, s – cadena, w – longitud del lado derecho que se cortará.
	
	Ejemplo:
	
	w=(6);//longitud
	
	s=("get without right *side* of *string*");//cadena
	
	y=strnright(s,w);
	
	Eventorecibido:
	
	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<get without right *side* of *string*>,name<s>,time<15:44:31>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value< get without right *side*>,name<y>,time<15:44:31>,date<24-09-04>

get_substr[3]	Obtener subcadena(1 –cadena, 2 – subcadena con la que iniciar, 3 – subcadena con la que finalizar, "\r" –final de la cadena)
	
	Formato: y=get_substr(s,w,x); donde y –valor(subcadena), s – cadena, w –subcadena con la que iniciar, x–subcadena con la que finalizar("\r" – final de la cadena)
	
	Formato de grabación: <NN.NN>
	
	Ejemplo:
	
	s=("getsubstring 1234567890");//cadena
	
	w=("to");// subcadena con la que iniciar
	
	x=("\r");//subcadena con la que finalizar, "\r" –final de la cadena
	
	y=get_substr(s,w,x);
	
	Eventorecibido:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<get substring 1234567890>,name<s>,time<16:34:13>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<sub>,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<substring 1234567890>,name<y>,time<16:34:13>,date<24-09-04>
	
	Ejemplo:
	
	s=("getsubstring 1234567890");//cadena
	
	w=("to");// subcadena con la que iniciar
	
	x=(1);//subcadena con la que finalizar, "\r" –final de la cadena
	
	y=get_substr(s,w,x);
	
	Eventorecibido:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<get substring 1234567890>,name<s>,time<16:36:26>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<sub>,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<substring >,name<y>,time<16:36:26>,date<24-09-04>

strltrim[1]	Borrar espacios a la izquierda
	
	Formato: y=strltrim(w); donde y –valor de la cadena de resultado, w – cadena.
	
	Ejemplo:
	
	w=(" remove spaces on the left");//cadena
	
	y=strltrim(w);
	
	Evento recibido:
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value< remove spaces on the left>,name<w>,time<17:07:49>,date<24-09-04>
	
	Event : CORE VAR_CHANGED int_obj_id<1>,value<remove spaces on the left>,name<y>,time<17:07:49>,date<24-09-04>

strrtrim[1]	Borrar espacios a la derecha
	Formato: y=strrtrim(w); donde y–valor de la cadena de resultado, w –cadena.
	Ejemplo:
	w=("Remove spaces on the right ");//cadena
	y=strrtrim(w);
	Evento recibido:
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Remove spaces on the right >,name<w>,time<17:18:35>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<Remove spaces on the right>,name<y>,time<17:18:35>,date<24-09-04>

stratrim[1]	Borrar espacios a ambos lados
	Formato: y=stratrim(w); donde y –valor de la cadena de resultado, w– cadena.
	Ejemplo:
	w=(" remove spaces on both sides ");//cadena
	y=stratrim(w);
	Evento recibido:
	Event : CORE VAR_CHANGED int_obj_id<1>,value< remove spaces on both sides >,name<w>,time<17:27:44>,date<24-09-04>
	Event : CORE VAR_CHANGED int_obj_id<1>,value<remove spaces on both sides>,name<y>,time<17:27:44>,date<24-09-04>

Nota.
Las funciones de fecha<DD-MM-YY>y hora<HH:MM:SS>devuelven la fecha y hora actuales. La función pi <3,14159265358979323846264338327 95>devuelve el valor de π.

Ejemplos de scripts

En la página:

A continuación, se pueden encontrar algunos ejemplos de scripts.

Ejemplo 1.

Mostrar cámara activa en la pantalla del monitor análogo.

Implementación:

```
OnEvent ("MONITOR","1","ACTIVATE_CAM")
{
   DoReact("CAM",cam,"MUX1");
}
```

Ejemplo 2.

Iniciar y detener vigilancia PTZ mediante macros.

Implementación:

```
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>");
}
```

Ejemplo 3.

Mostrar en pantalla una cámara de alarma en el modo de monitor único.

Implementación:

```
OnEvent ("CAM",N,"MD_START")
{
   DoReact("MONITOR","1","ACTIVATE_CAM","cam<"+N+">");

   DoReact("MONITOR","1","KEY_PRESSED","key<SCREEN.1>");
}
```

Ejemplo 4.

Aquí se muestra un ejemplo de bucle infinito y la forma de detenerlo. La macro1 inicia el bucle y la macro2 lo detiene.

Implementación:

```
OnEvent("MACRO","1","RUN") //la macro1 se ha ejecutado

{
   //se necesitan corchetes para aislar la sentenciade espera en cada flujo individual
   [
   flag=1;
   for(a=1;flag<2;a=1) //sentencia de ciclo
   {
      Sleep(500); //la sentencia de espera crea una pausa de 500 milisegundos
      ff="!!!!!!!!!!!!!!!!!!";
   }
   ]
}

OnEvent("MACRO","2","RUN") // la macro2 se ha ejecutado
{
   flag=2;
}
```

Ejemplo 5.

Un monitor de alarma de video de la cámara de alarma está siempre encendido.

Implementación:

```
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+">");
   }
}
```

Ejemplo 6.

Se reproducirá un archivo de audio a partir del momento en que aparece un evento y finalizarácuando un nuevo evento aparezca.(Una macro se ejecuta en este caso).

> ⚠ **¡¡¡Importante!!!**
> Un archivo de audio no debe ser mayor que el número de segundos especificado en el operador Wait.

Implementación:

```
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;
}
```

Ejemplo 7.

Hay 2 cámaras con dispositivos PTZ. Una cámara debe rotar a la posición predeterminada y captar una instantánea cada 15 minutos. La hora actualizada es el nombre del archivo.

Implementation:

```
OnTime(W,D,X,Y,H,M, "01")

{
   if(strequal(M,"0"))
   {
      name=H+"_"+M+"_"+S+".jpg";
      //Dispositivo PTZ 1.1 de la cámara 1
      name1="Camera1"+name;
      DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
      DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
      //Dispositivo PTZ 1.2 de la cámara 2
      name="Camera2"+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";
      //Dispositivo PTZ 1.1 de la cámara 1
      name1="Camera1"+name;
      DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
      DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
      //Dispositivo PTZ 1.2 de la cámara 2
      name="Camera2"+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";
      // Dispositivo PTZ 1.1 de la cámara 1
      name1="Camera1"+name;
      DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
      DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
      //Dispositivo PTZ 1.2 de la cámara 2
      name="Camera2"+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";
      // Dispositivo PTZ 1.1 de la cámara 1
      name1="Camera1"+name;
      DoReact("TELEMETRY","1.1","GO_PRESET","preset<1>,tel_prior<1>");
      DoReact("MONITOR","1","EXPORT_FRAME","cam<1>,file<d:\"+name1);
      //Dispositivo PTZ 1.2 de la cámara 2
      name="Camera2"+name;
      DoReact("TELEMETRY","1.2","GO_PRESET","preset<1>,tel_prior<1>");
      DoReact("MONITOR","1","EXPORT_FRAME","cam<2>,file<d:\"+name);

   }

}
```

Ejemplo 8.

El audio que proviene de un micrófono (OLXA_LINE) no se registra simultáneamente con una cámara. Por defecto, el micrófono no está conectado.El audio se grabará cuando se active mediante sonido y la cámara lo detecte.

> **Nota.**
> Los comandos RECORD_START y RECORD_STOP se han añadido al micrófono en la versión 4.7.0 y posteriores.

La grabación de audio empieza con un arranque acústico (ACCU_START) y la detección de movimiento (MD_START) y el indicador de variable se incrementa en 1. Cuando el arranque acústico y la detección de movimiento finalizan, el indicador de variable se reduce en 1 y la grabación de audio se detiene sólo si es 0, es decir, que no existe arranque acústico o movimiento.

Implementación:

```
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");
   }
}
```

Ejemplo 9.

El sistema cuenta con varias cámaras (núm).Se debe comprobar el funcionamiento de la detección de movimiento en todas las cámaras (también se puede utilizar para comprobar los sensores de seguridad).

Para resolver este problema, se puede utilizar la emulación de un conjunto lineal de símbolos (cadena), es decir, se completa una matriz de símbolos (“N” en este caso). Cuando se activa la detección de movimiento en la cámara, el elemento correspondiente de la matriz (ID de la cámara) cambia (a “Y”). Así, obtenemos una matriz de símbolos “N” (que no se habían activado) y de “Y” (que se había activado) como resultado.

Se contabiliza el número de activaciones y el mensaje con el número total de cámaras y el número de cámaras en las que la herramienta de detección se activó. La comprobación se inicia mediante la Macro1 y se detiene con la Macro2.

Implementación:

```
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="Total:"+str(num)+"Triggered:"+str(fin);
   rez=MessageBox("",tmp,0);

}
```

Ejemplo 10.

Vigile varias zonas utilizando accionespredefinidas de dispositivos PTZ; la detección de movimiento se debe activar en alguna área de estas zonas.

Cámara1. 5 zonas de detección y 5 acciones predefinidas. Estos parámetros se configuran con la variable *n*. La Macro1 inicia la ejecución del algoritmo. La Macro2 finaliza la ejecución del algoritmo. Indicador – variable interna.

Cuando el algoritmo se inicia, la cámara cambia a predefinido1 y activa la primera zona de detección.El retraso entre estos comandos es de 200 milisegundos (para que la cámara cambie a predefinido). Transcurridos 5 segundos, la 1ª zona se desactiva y el bucle empieza de nuevo, pero con la 2ª zona y predefinido2. Y así sucesivamente hasta que vuelve a empezar desde el inicio. El algoritmo se detiene si el indicador de variable se restablece a cero (mediante la Macro2).

Implementación:

```
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;
}
```

Ejemplo 11.

Tenemos 2 pantallas – la primera muestra un monitor virtual con cámaras; la segunda muestra el objeto **Mapa** con sensores *Bolid*. Cuando la cámara hace saltar la alarma – se muestra la pantalla 1; cuando el sensor activa la alarma – se muestra la pantalla 2, pero solamente en el CLIENTE REMOTO.

Implementation:

```
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 >");
}
```

Ejemplo 12.

Añada subtítulos al video de la cámara 1 cuando un evento de alarma aparezca en esta cámara. Cuando el evento de alarma finalice, añada subtítulos que indiquen el fin de la alarma.

Implementación:

```
OnEvent("CAM","1","MD_START")
{
  DoReact("CAM","1","CLEAR_SUBTITLES","title_id<1>"); //borrar todos los subtítulos del video
  DoReact("CAM","1","ADD_SUBTITLES","command<Camera 1 Alarm "+ time +
"\r>,page<BEGIN>,title_id<1>"); //el parámetro time añade la hora del evento registrándola en
los subtítulos
}
OnEvent("CAM","1","MD_STOP")
{
  DoReact("CAM","1","ADD_SUBTITLES","command<Camera 1 Alarm end "+ time +
"\r>,page<END>,title_id<1>");
}
```

Nota
Cuando se estén utilizando los parámetros de página <BEGIN> y página <END>, se rellenan los campos correspondientes de la base de datos de los subtítulos– esto permite la búsqueda de datos a través del objeto de interfaz **Búsqueda por títulos**.

Descripción de las reacciones de los objetos de sistema

Todas las reacciones de los principales objetos del sistema quedan descritas en esta sección.

Nota.
Se pueden visualizar los eventos para objetos de sistema mediante una de las siguientes formas:

1. Visualizando el archivointellect.ddi mediante el uso de la utilidad ddi.exe (ver el Paquete de Software Intellect. Guía del Administrador).
2. Visualizandolos eventos del objeto de sistema seleccionado por medio del panel de control del objeto de sistema **Macro** (ver Paquete de Software Intellect. Guía del Administrador).

GRABBER

El objeto **GRABBER** corresponde al objeto de sistema de **Dispositivo de Captura de Video**.

El objeto **Grabber** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento correspondiente.

Procedimiento de formato de eventos para el dispositivo de captura de video:

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

Descripción de los eventos del objeto **Grabber**.

Evento	Descripción
"+12V"	Error de voltaje +12V.
"+3.3V"	Error de voltaje+3.3V.
"+5V"	Error de voltaje+5V.
"-12V"	Error de voltaje-12V.
"-5V"	Error de voltaje-5V.
"CPU_FAN"	Número de rotaciones del ventilador.
"CPU_TEMP"	Temperatura del procesador.
"SYS_TEMP"	Temperatura del chipset MB.
"UPS_COMMLOST"	Pérdida de conexión.
"UPS_FATAL_ERROR"	Error de conexión.

"UPS_LOWBATT"	Batería baja.
"UPS_ONBATT"	Cambiar a suministro de batería.
"UPS_ONLINE"	Restaurar el suministro principal.
"UPS_REPLACEBATT"	Se necesita cambio de batería.
"UPS_SHUTTING"	Apagar.
"VCORE"	Voltaje del núcleo del procesador.
"AUDIO_SIG_LOST "	Pérdida de sonido
"CONNECT_FAIL"	Error de conexión
"CONNECT_OK "	Conectado
"NETWORK_FAILURE "	Pérdida de conexión
"STATE_CONNECTED "	Conexión restablecida

Éste es el formato de operador para describir las acciones con el dispositivo de captura de video:

```
DoReact("GRABBER","_id_","_comando_"[,"_parámetros_"]);
```

En la siguiente tabla se presenta la lista de comandos y parámetros del objeto **Grabber**:

Comando – descripción del comando	**Parámetros**	**Descripción de los parámetros**
"SETUP" – configura los parámetros del dispositivo de captura de video.	chan<>	Número de ranuras PCI (0,1,2,…,32).
	mode<>	Velocidad de captura/digitalización (0 – máxima, 1 – media, 2 – mínima).
	resolution<>	Resolución (0– estándar, cuarto de cuadro (384x288); 1 – alta, medio cuadro (768x288); 2 – marco máximo (768x576)).
	format<>	Formato de la señal de video (PAL, NTSC).
	drives<>	Discos para la grabación de archivos de video (DRIVE1:\, DRIVE2:\ … DRIVEN:\).
	cams<>	Número de cámaras de video conectadas
	auth<>	
	ip<>	Dirección IP de la tarjeta de red de entrada de video
	name<>	Nombredelobjecto.
	flags <>	Indicadores.
	ip_port<>	Puerto IP.
	password<>	Contraseña.
	type<>	Tipo de digitalización.
	username<>	Acceso.
	watchdog<>	Cierre de WatchDog (0 –desactivado, 1 – activado).

"SET_DRIVES" – configura los discos para la grabación de archivos de video.	drives<>	Discos para la grabación de archivos de video.
"MUX1_OFF" – desactiva la salida de video mediante la salida analógica 1.	-	-
"MUX2_OFF" - desactiva la salida de video mediante la salida analógica 2.	-	-
"MUX3_OFF" - desactiva la salida de video mediante la salida analógica 3.	-	-
"SET_IPINT_PARAM" – Configura (cambia) los parámetros del dispositivo IP. La reacción permite cambiar la configuración del dispositivo IP sin entrar en su interfaz web. *Nota. Para la operación de reacción es necesario activar el modo de señal de video de flujo múltiple - ver Guía del Administrador, sección Configuración de video en flujo múltiple y el* Apéndice 2. Definición de los valores param_id y param_value para la reacción SET_IPINT_PARAM	param_id<>	Nombre del parámetro. El conjunto de parámetros para cada cámara es individual – ver Apéndice 2. Definición de los valores param_id y param_value para la reacción SET_IPINT_PARAM
	param_value<>	Valor del parámetro. El conjunto de parámetros para cada cámara es individual – ver Apéndice 2. Definición de los valores param_id y param_value para la reacción SET_IPINT_PARAM
	cam_id<>	ID de la cámara en el paquete de software*Intellect*.
	vstream_id<>	El número de flujos de video (parámetro opcional) se da como "Número de cámara"."Número de flujo", por ejemplo, 1.1, 1.2.

Las propiedades del objeto **GRABBER** quedan reflejadas en la tabla.

Propiedades del objetoGRABBER	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	Número del dispositivo de captura de video.

Ejemplos de uso de eventos y reacciones del objeto **Dispositivo de captura de video**:

1. Se necesita establecer el primer canal en el primer dispositivo de captura de video, a máxima velocidad de digitalización, resolución a medio cuadro y formato PAL mientras se inicia la primera macro.

   Nota.
   La descripción del objeto MACRO es la siguiente (ver la sección MACRO).

```
OnEvent("MACRO","1","RUN")
// iniciarmacro 1
{
   DoReact("GRABBER","1","SETUP","chan<1>,mode<0>,resolution<1>,format<PAL>");

   //establecer el canal 1 en el primer dispositivo de captura de video, la velocidad de la digitalización es la máxima, la resolución es medio cuadro, el formato es PAL
}
```

2. Configure los discos D:\ y F:\ para la grabación de archivos de video mientras se inicia la tercera macro.

```
OnEvent("MACRO","3","RUN") //iniciar macro 3
{
   DoReact("GRABBER","1","SET_DRIVES","drives<D:\,F:\>"); //grabar el archivo de video en los discos D:\ y F:\
}
```

3. Se necesita mostrar en pantalla la primera cámara de video en la primera salida analógica y desactivar las primeras salidas analógicas de la primera y segunda tarjetas mientras se produce un error de conexión en el segundo dispositivo de captura de video.

```
OnEvent("GRABBER","2"," UPS_FATAL_ERROR") //error de conexión en el dispositivo de captura de video 2
{
   DoReact("CAM","1","MUX1"); //mostrar la cámara de video 1 en la 1-era salida analógica de la tarjeta
   Wait(5);
   DoReact("GRABBER","1","MUX1_OFF"); //desactivar 1-era salida analógica de la primera tarjeta
   DoReact("GRABBER","2","MUX1_OFF"); // desactivar 1-era salida analógica de la segunda tarjeta
}
```

> **Nota.**
> Si las salidas analógicas de dos o más tarjetas están conectadas en paralelo y la cámara de video 1 pertenece al primer captador y la cámara de video 2 pertenece al segundo captador, mientras ejecuta el comando «DoReact("CAM","1","MUX1");» es necesario ejecutar el comando «DoReact("GRABBER","2","MUX1_OFF");» primero, y de la misma forma, mientras ejecuta el comando «DoReact("CAM","2","MUX1");» es necesario ejecutar el comando «DoReact("GRABBER","1","MUX1_OFF");» primero. De otro modo, se producirá una superposición de señales.

> **Nota.**
> La descripción del objeto **CÁMARA** es la siguiente (ver la sección CAM).

4. Es necesario desactivar la segunda salida analógica del dispositivo de captura de video mientras se restaura el suministro principal.

```
OnEvent("GRABBER","1","UPS_ONLINE")//restaurando el suministro principal
{
   DoReact("GRABBER","1","MUX2_OFF");//desactivar la salida analógica 2
}
```

CAM

El objeto **CAM** se corresponde con el objeto de sistema **Cámara**.

El objeto **CAM** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento correspondiente.

Formato del procedimiento de los eventos en el objeto **Cámara**:

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

Descripción de los eventos del objeto **CAM**.

Eventos	Descripción	Comentario

"ARM"	La cámaraestá armada.	
"ATTACH"	Conectanda.	
"BLINDING"	La cámara está sellada.	
"DETACH"	Corte	El evento se genera cuando se produce una pérdida de la señal de entrada de la cámara en el dispositivo de captura de video.
"DISARM"	La cámara está desarmada.	
"FILE_REC_ERROR"	Error de registro en disco.	El evento se genera cuando ocurre un error en el registro del archivo de video en el disco.
"MD_START"	Alarma.	
"MD_STOP"	Fin de la alarma.	
"PRINT"	Imprimir cuadro.	
"REC"	Grabar en disco.	
"REC_STOP"	Detener la grabación en disco.	
"UNBLINDING"	La cámara está abierta.	
"RECORDER_ON"	La grabación está activada.	
"RECORDER_OFF"	La grabación está desactivada.	
"DISC_MOUNT"	El disco está montado.	
"DISC_UNMOUNT"	El disco está desmontado.	
"FINISHED_AVI_EXPORT"	La exportación de video está completa.	Si la exportación de video falla, el evento contiene un parámetro vacío error_result.

El formato del operador para describir acciones con la cámara es:

```
DoReact("CAM","_id_","_comando_" ["_parámetros_"]);
```

La lista de comandos y parámetros para el objeto **CÁMARA**aparecen en la siguiente tabla:

Comando – descripción del comando	**Parámetros**	**Descripción de los parámetros**
"SETUP" – configurar (modificar) los parámetros de cámara	rec_priority<>	Prioridad de grabación (de 0 a 3, 0 – estándar, 3 – con todos los recursos).
	compression<>	Racionamiento de la compresión (0 – sin compresión, 1- calidad máxima, …, 5 – calidad mínima).
	sat_u<>	Valor del color (0 – mín, 10 – máx).
	proc_time<>	Periodo de agregación (0 – 30 seg).
	manual<>	Control de brillo y ajuste de contrastes (0 – manual; 1 – auto; 2 – auto, pero cerca de los valores especificados manualmente).
	contrast<>	Contraste (0 – mín, 10 – máx).
	md_size<>	Tamaño de los objetos en detección de movimiento (1 -16).

	md_mode<>	Modo de registro de pausas (1 – activado, 0 desactivado).
	audio_type<>	Tipo de acompañamiento acústico.
	pre_rec_time<>	Tiempo de pre-registro (0 – 20 seg).
	bright<>	Brillo (0 – mín, 10 – máx).
	audio_id<>	Número de micrófono (parámetro vacío si no existe micrófono).
	rec_time<>	Velocidad de grabación (1 – 30 fps, 0 – sin uso).
	alarm_rec<>	Registro de alarmas (1 – activado, 0 – desactivado).
	hot_rec_time<>	Tiempo de registro en caliente (0 – 30 seg).
	hot_rec_period<>	Periodo de registro en caliente (0 – 20 seg).
	mux<>	Número de canal (0 – 1 canal, 15 – 16 canal).
	color<>	Color (0 – blanco y negro, 1 – multicolor).
	activity<>	-
	arch_days<>	Número de días de archivo.
	blinding<>	La cámaraestásellada.
	config_id<>	-
	decoder<>	-
	flags<>	Indicadores.
	fps<>	Velocidad de grabación (0 – sin uso, 1 – 30 fps).
	ifreq<>	Frecuencia de cuadros de anclaje en secuencia (1 – todos son cuadros de anclaje, 2 – 100 cuadros).
	mask 0, mask1, mask2, mask3, mask4	Máscara de detección.
	md_contrast<>	Sensibilidad de detección de movimiento (0 – 15).
	motion<>	Estimación del compresor de movimiento (5 - 255).
	name<>	Nombre del objeto.
	password_crc<>	Contraseña para el archivo de video.
	priority<>	Prioridad del recurso de registro (0 – auto, 1 – manual).
	resolution<>	Resolución (0 – estándar CIF, 1 - alta 2CIF, 2 –máxima 4CIF).
	type<>	Tipo de objeto.
	yuv<>	Esquema de color del codificado de la señal de video (0 – YUV4:2:0, 1 - YUV4:2:2).
"DELETE" – desactivar la cámara.	-	-
"START_VIDEO" – activar el flujo de video de la cámara en uso.	slave_id<>	Nombre del equipo al que la cámara está conectada.

	comress<>	Valor de compresión.
	register_only<>	-
"STOP_VIDEO" – desactivar el flujo de video de la cámara en uso.	slave_id<>	Nombre del equipo al que la cámara está conectada.
"REQUEST_MASK"	mask<>	Máscara.
"MUX1", "MUX2", "MUX3" – mostrar la imagen de la cámara en las salidas analógicas 1, 2, 3.	-	-
"ACTIVATE" – mostrar la cámara en el monitor.	monitor<>	Número de monitor.
"ARM" – activar la cámara.	-	-
"DISARM" – desactivar la cámara.	-	-
"REC" – iniciar la grabación de la cámara.	time<>	Tiempo de grabación en segundos, si es nulo sólo se graba un cuadro.
	rollback<>	Si es uno, la grabación se realiza con un retroceso.
	priority<>	Establece la prioridad del comando para iniciar la grabación. Ver Apéndice 1. Prioridades del inicio y detención Prioridades de inicio y detención de los comandos de grabación
"REC_STOP" – detener la grabación de la cámara.	priority<>	Establece la prioridad del comando para detener la grabación. Ver Apéndice 1. Prioridades de inicio y detención de los comandos de grabación
"SET_MASK" – establecer máscara.	mask<>	Máscara.
"ADD_SUBTITLES" – añadir títulos.	command<>	Test de títulos impuestos.
	title_id<>	ID del objeto **Subtitulador** utilizado para la imposición.
	page<>	El parámetro permite registrar títulos en la base de datos de títulos para proporcionar búsquedas por título. Valores disponibles: BEGIN (iniciar el registro en la base de datos), END (finalizar el registro en la base de datos).
"SIP_CONNECT" - Sipconectada	-	-
"SIP_DISCONNECT" – Sipdesconectada	-	-
"SET_IPINT_PARAM" – Configurar (modificar) los parámetros del dispositivo IP. La reacción permite cambiar la configuración del dispositivo IP sin entrar en su interfaz web. *Nota. Para la operación de reacción se necesita activar el modo de señal de video de flujo múltiple – ver Guía del Administrador, sección Configuración de video en transmisión de flujo múltiple, y Apéndice 2. Definición de los valores* param_id *y* param_value *para la reacción* SET_IPINT_PARAM	param_id<>	Nombre del parámetro. El conjunto de parámetros para cada cámara es individual - ver *Apéndice 2. Definición de los valore s* param_id *y* param_value *para la reacción* SET_IPINT_PARAM
	param_value<>	Valor del parámetro. El conjunto de parámetros para cada cámara es individual - ver *Apéndice 2. Definición de los valores* param_id *y* param_value *para la reacción* SET_IPINT_PARAM
	vstream_id<>	Número de flujo de video (parámetro opcional). Se determina mediante "Número de cámara"."Número de flujo", por ejemplo, 1.1, 1.2.
GET_FRAME – obtener cuadro de una cámara incluso a pesar de que no aparezca en pantalla en el monitor de video vigilancia.	path<>	Ruta para guardar el cuadro. Si no existe ningún parámetro, se creará en el sistema el evento FRAME_SENT con el parámetro de datos. El procesamiento de este evento queda descrito en la sección El método SaveToFile (guardar en archivo) de la Guía de Programación (JScript).

Las propiedades del objeto **«CAM»** quedan reflejadas en la tabla.

Propiedades del objeto«CAM»	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto principal.
TELEMETRY_ID<>	ID del módulo de telemetría (ID deptz).
REGION_ID<>	ID de la región.

Ejemplos de uso de eventos y reacciones del objeto **Cámara**:

1. Cambiar la cámara a modo color e iniciar la grabación desde ésta mientras se activa la primera cámara.

```
OnEvent("CAM","1","ARM") //la cámara de video 1 está activada
{
   DoReact("CAM","1","SETUP","color<1>"); // establecer modo color de la cámara de video
   DoReact("CAM","1","REC"); //grabar desde la cámara 1
}
```

2. Activar la primera cámara de video mientras se desactiva la quinta cámara de video.

```
OnEvent("CAM","5","DETACH") // la cámara de video 5 está desactivada
{
   DoReact("CAM","1","ARM"); //la cámara de video 1 está activada
}
```

3. Utilizar la mitad de los recursos mientras hay una grabación en curso de la primera cámara (i.e. si hay 4 cámaras de video conectadas mediante el dispositivo de captura, la primera cámara grabará a una velocidad de 6 fps y,las tres restantes, a 2-2,5 fps) si ésta se encuentra en estado de alarma.

```
OnEvent("CAM","1","MD_START") //cámara 1 se encuentra en estado de alarma
{
   DoReact("CAM","1","SETUP","rec_priority<2>"); // utilizar la mitad de los
recursosmientrasestágrabando
}
```

4. Establecer la máxima compresión en sincronización con el cuarto micrófono de la tarjeta de audio en la primera cámara de video mientras la primera cámara de video está realizando una grabación en disco.

```
OnEvent("CAM","1","REC") //cámara de video 1 grabando en disco
{
   DoReact("CAM","1", "SETUP", "compression<5>, audio_type<OLXA_LINE>,
audio_id<4>"); //cámara de video 1, compresión máxima, en sincronización con micrófono
4 de la tarjeta de audio
}
```

5. Iniciar la grabación desde la primera cámara a calidad mínima en modo blanco y negro cuando ésta deje de estar en estado de alarma.

```
OnEvent("CAM","1","MD_STOP") // cámara 1 ha dejado de estar en estado de alarma
{
   value = 5;
   DoReact("CAM", "1", "SETUP", "compression<" + value + ">,color<0>");
   //iniciar grabación desde la cámara de video 1 con calidad mínima en modo Blanco y
negro.
}
```

6. Iniciar la grabación desde la primera cámara en modo retroceso (“rollback”) cuando ésta se encuentre desactivada.

```
OnEvent("CAM","1","DISARM")  //cámara de video 1 está desactiva
{
   DoReact("CAM","1","REC","rollback<1>");  // Iniciar grabación desde la cámara de Video
1 en modo “retroceso”
}
```

7. Configurar parámetros nuevos de señal de video mientras se está conectando la primera video cámara.

```
OnEvent("CAM","1","ATTACH") //cámara de video 1 está conectada
{
   VIDEO_CANAL_ID = GETOBJECTPARAM("CAM","1","PARENT_ID"); // definir ID del canal
de video al que pertenece la cámara

DoReact("GRABBER",VIDEO_CANAL_ID,"SETUP","chan<0>,mode<0>,resolution<1>,format
<pal>"); //configurar nuevos parámetros del canal de video.
}
```

8. Iniciar giro automático en la Cámara 1 cuando la Macro 2 se ha ejecutado.

```
OnEvent
  ("MACRO","2","RUN")
{

DoReact("CAM","1","CRUISE_START","cruise_id<1>,action<CRUISE_START>,cam_id<1>");
}
```

Comprobar el estado de la función del objeto **CAM**:

```
CheckState("CAM","number","state")
```

El objeto **CAM** puede encontrarse en los siguientes estados.

Estado del objeto «CAM»	**Descripción**
"ALARMED"	La cámara está en modo alarma.
"DISARM_DETACHED"	La cámara no tiene señal.
"DETACHED"	La cámara no tiene señal.
"ARMED"	La cámara está activada.
"DISARMED"	La cámara está desactivada.

PLAYER

El objeto **PLAYER** corresponde al objeto de sistema **Reproductor de Audio**.

El formato de operador para caracterizar acciones con el reproductor de audio es el siguiente:

```
DoReact("PLAYER","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla presenta la lista de comandos y parámetros del objeto **REPRODUCTOR**.

Comando – descripción del comando	**Parámetros**	**Descripción**
"PLAY_WAV" – reproduce un archivo de audio.	file<>	Archivo de audio con la ruta completa hacia él.
"SETUP" – configuración de los parámetros del reproductor de audio.	board<>	Unidad de sonido del reproductor de audio.
	flags<>	Indicadores.
	h<>	Alto del diálogo de notificaciones (0 – 100).
	name<>	Nombre del objeto.
	voice<>	Notificación acústica.
	voice_board<>	Unidad de sonido de notificaciones.
	w<>	Ancho del diálogo de configuraciones (0 – 100).
	x<>	Esquina superior izquierda del diálogo de configuraciones (0 – 100).
	y<>	Esquina superior izquierda del diálogo de configuraciones (0 – 100).

La siguiente tabla describe las propiedades del objeto **PLAYER**.

Propiedades del objeto REPRODUCTOR	**Descripción de las propiedades**
ID<>	ID de objeto.
PARENT_ID<>	ID del objetomatriz.

Ejemplos de utilización de eventos y reacciones del objeto **Reproductor de Audio**:

1. Reproducir el archivo de audio con la dirección «C:\ Program Files\Intellect\Wav\cam_alarm_1.wav» al activar el indicador de funcionamiento del reproductor de audio.

```
OnEvent("PLAYER","1","flags") // activar el indicador de funcionamiento del reproductor de audio.
{
   DoReact("PLAYER","1","PLAY_WAV","file<C:\ program files\intellect\wav\cam_alarm_1.wav >");
// reproducir archivo de audio
}
```

DIALOG

El objeto **DIALOG** corresponde al objeto de sistema **Panel de consultas de operador**:

El formato de operador que describe las acciones del panel de consultas de operador es:

```
DoReact("DIALOG","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla presenta la lista de comandos y parámetros para el objeto **DIÁLOGO**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"SETUP" – configuración del panel de consultas de operador.	x<>	Coordenada de la esquina superior izquierda(0 - 100).

	y<>	Coordenada de la esquina superior izquierda(0 - 100).
	allow_move<>	0 – impedir movimiento, 1 – permitir movimiento.
"RUN" – mostrar el panel de consultas de operador.	-	-
"RUN_MODAL" – ejecutar el panel de consultas de operador en modo modal.	-	-
"CLOSE" – cerrar el último panel de consultas de operador abierto.	-	-
"CLOSE_ALL" – cerrar todos los paneles de consultas de operador abiertos.	-	-

Ejemplos de uso de reacciones del objeto **Panel de consultas de operador**:

1. Utilizar la macro 1 para establecer las coordenadas de la esquina superior izquierda del panel de consultas de operador (cámara de video PTZ PANASONIC-850) en el centro de la pantalla, impedir el movimiento de éste, y mostrarlo en pantalla.

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

2. Cerrar el panel de consultas de operador utilizando la Macro 2.

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

MMS

El objeto **MMS** corresponde al objeto de sistema **Servicio de Mensajes de Correo**.

El objeto **MMS** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento correspondiente.

Formato del procedimiento de los eventos en el objeto **Servicio de Mensajes de Correo**:

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

Evento	**Descripción**
"SET_CONNECTIONS"	Listado de conexiones disponibles.

El formato de operador que describe las acciones con el servicio de mensajes de correo es el siguiente:

```
DoReact("MMS","_id_","_comando_" ["_parámetros_"]);
```

List of commands and parameters for the MMS object is given in the table:

Comando – descripción del comando	**Parámetros**	**Descripción**

"SETUP" – configuración del servicio de mensajes de correo.	smtp<>	Dirección del servidorSMTP.
	connection<>	Tipo de conexión.
	smtp_username<>	Nombre de usuario.
	smtp_password<>	Contraseña.
	port<>	Número de puerto.
	flags<>	Indicadores.
	name <>	Nombre del objeto.
"GET_CONNECTIONS" – obtener el listado de conexiones disponibles.	-	-

La siguiente tabla muestra las propiedades del objeto **MMS**:

Propiedades del objeto MMS	**Descripción**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

Ejemplo de utilización de reacciones del objeto **Servicio de Mensajes de Correo**.

1. Definir el número de puerto 25 como servidor de mensajes de correo al activar la macro 1.

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

MAIL_MESSAGE

El objeto **MAIL_MESSAGE** corresponde al objeto de sistema **Mensaje de correo**.

El objeto **MAIL_MESSAGE** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento correspondiente.

Formato del procedimiento de los eventos en el objeto **Mensaje de correo**:

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

Evento	**Descripción**
"SEND_ERROR"	Error al enviar el mensaje.
"SENT"	El mensaje ha sido enviado.

El formato de operador para describir las acciones con el objeto mensaje de correo es el siguiente:

```
DoReact("MAIL_MESSAGE","_id_","_comando_" [,"_parámetros_"]);
```

La siguiente tabla define la lista de comandos y parámetros del objeto **MAIL_MESSAGE**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"SETUP" – configuración del objeto mensaje de correo.	from<>	Dirección de origen.
	to<>	Dirección de destino.
	cc<>	Copias.

	subject<>	Asunto del mensaje.
	body<>	Cuerpo del mensaje.
	attachments<>	Adjuntos. Si se adjuntan varios archivos, se deben separar las direcciones con punto y coma.
	flags<>	Indicadores.
	name<>	Nombre del objeto.
	pack<>	Método para agruparadjuntos.
"SEND" – enviar mensaje de correo.	-	-

Las propiedades del objeto **MAIL_MESSAGE** se muestran en la tabla.

Propiedades del objeto MAIL_MESSAGE	**Descripción**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

Ejemplo de utilización de reacciones del objeto **Mensaje de correo**.

1. Enviar mensaje con imagen de la cámara de video cuando ésta cambie su estado a alarma al mismo tiempo que se activa la detección de movimiento.

```
OnInit()
{
   i=0; //el Contador se usa para impedir la sobrescriturade imágenes de una cámara
}

OnEvent("CAM",N,"REC") //la cámara de video se encuentra en estado de alarma
{
   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<camera is triggered"+ N + ">,
subject<alarm by camera>, from<sergey.kozlov@itv.ru>,
to<sergey.kozlov@itv.ru>,attachments<" + filename + ">");
   DoReact("MAIL_MESSAGE","1","SEND");
}
```

VMS

El objeto **VMS** corresponde al objeto de sistema **Servicio de Mensajes de Voz**.

El formato operador que describe las acciones del servicio de mensajes de voz es:

```
DoReact("VMS","_id_","_comando_" [,"_parámetros_"]);
```

La siguiente tabla muestra la lista de comandos y parámetros para el objeto **VMS**:

Comando – descripción del comando	**Parámetros**	**Descripción de los parámetros**
"SEND" – enviar mensaje.	modem<>	Nombre del dispositivo.
	pulse<>	Tipo de marcación (0 – tonal, 1 – por pulsos).
	name<>	Nombre del objeto.
	redial_attempts<>	Número de intentos de llamada.

	redial_delay<>	Pausa entre intentos de llamada.
	waitfordialtone<>	Esperando señal de línea (0 – no, 1 – sí).
	flags<>	Indicadores.

La siguiente tabla muestra las propiedades del objeto **VMS**.

Propiedades del objetoVMS	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

Ejemplos de uso de eventos y reacciones del objeto **Servicio de Mensajes de Voz**:

1. Se necesita enviar un mensaje al ejecutar la macro 1 si el módem está conectado al Puerto COM2, el tipo de marcación es tonal, sin esperar a la señal de tono.

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

GRELE

El objeto **GRELE** corresponde al objeto de sistema **Relé**.

El objeto **GRELE** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el correspondiente evento.

El formato de procedimiento de los eventos para el relé es:

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

Evento	**Descripción**
"OFF"	ReléApagado.
"ON"	ReléEncendido.
"SIGNAL_LOST"	Pérdida de señal.

El formato operador que describe las acciones con el dispositivo de captura de video es:

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

La siguiente tabla muestra la lista de comandos y eventos para el objeto **GRELE**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"ON" –activarrelé.	-	-
"OFF" –desactivarrelé.	-	-
"SETUP" – configuración del relé.	chan<>	Número de salida (0 – 15).
	flags<>	Indicadores.
	name<>	Nombre del objeto.

La siguiente tabla indica las propiedades del objeto **GRELE**.

Propiedades del objeto GRELE	**Descripción de las propiedades**
ID<>	ID del objeto.

PARENT_ID<>	ID del objeto matriz.
REGION_ID<>	ID de la Región.

Comprobar el estado de la función del objeto **GRELE**:

```
CheckState("GRELE","número","estado")
```

El objeto **GRELE** puede encontrarse en los siguientes estados:

Estado del objetoGRELE	Descripción del estado
"ON"	Relé ENCENDIDO.
"OFF"	ReléAPAGADO.
"DETACHED_ON"	Pérdida de la conexión.
"DETACHED_OFF"	Pérdida de la conexión.

Ejemplos de utilización de eventos y reacciones del objeto **Relé**:

1. Activar el relé 2 mientras se produzca una pérdida de conexión con el relé 1.

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

GRAY

El objeto **GRAY** se corresponde con el objeto de sistema **Sensor**.

El objeto **GRAY** envía los eventos reflejados en la tabla. El procedimiento se inicia en cuanto el evento correspondiente aparece.

El formato de procedimiento de los eventos para el sensor es:

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

Evento	Descripción
"ALARM"	Alarma. Este evento se recibe al abrir o cerrar el sensor (dependiendo de los ajustes del objeto) si el sensor está activado. Si el sensor está desactivado, se recibirán los eventos Sensor abierto y Sensor cerrado de acuerdo con la situación.
"ARM"	El sensorestáactivado.
"CONFIRM"	Alarmarecibida.
"DISARM"	El sensorestádesactivado.
"NOT_VALID_STATE"	La zona no está lista.
"OFF"	Sensor abierto.Este evento se recibe al abrirse el sensor si éste se encuentra desactivado.
"ON"	Sensor cerrado. Este evento se recibe al cerrar el sensor si éste se encuentra desactivado.
"SIGNAL_LOST"	Se ha perdido la conexión con el sensor.

El formatoperador que describe las acciones con el sensor es:

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

La siguiente tabla muestra la lista de comandos y parámetros para el objeto **GRAY**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"ARM" – activar sensor.	-	-
"DISARM" – desactivar sensor.	-	-
"CONFIRM" – confirmar alarma.	-	-
"SETUP" – configuración del sensor.	chan<>	Número de salida (0 – 15).
	flags<>	Indicadores.
	name<>	Nombre del objeto.
	type<>	Tipo de objeto sensor (0 – al cerrarse, 1 – al abrirse).

La siguiente tabla muestra las propiedades del objeto **GRAY**.

Propiedades del objetoGRAY	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.
REGION_ID<>	ID de la región.

Comprobar el estado de la función del objeto **GRAY**:

```
CheckState ("GRAY","número","estado")
```

El objeto **GRAY** puede encontrarse en los siguientes estados:

Estado del objeto GRAY	**Descripción del estado**
"ARMED"	El sensor está activado.
"DISARME""	El sensor está desactivado.
"ALARMED"	Alarma.
"CONFIRMED"	Alarmaconfirmada.
"DISARMED_ALARM"	Not ready.
"DETACHED_ARMED"	Pérdida de conexión.
"DETACHED_DISARM"	Pérdida de conexión.
"OFF"	Normal.

Ejemplos de utilización de eventos y reacciones del objeto **Sensor**:

1. Se debedirigir el segundo sensor a la segunda entrada si se pierde la conexión con el primer sensor.

```
OnEvent("GRAY","1"," SIGNAL_LOST") //se ha perdido la conexión con el primer sensor
{
   DoReact("GRAY","2","SETUP","chan<2>"); //el sensor está en la segunda entrada
}
```

2. Abrir el segundo sensor y activar grabación en retroceso de la primera cámara de video en el caso de que el primer sensor esté cerrado.

```
OnEvent("GRAY","1"," ON") //el primer sensor está cerrado
{
   DoReact("GRAY","2","SETUP","type<1>"); //abrir el segundo sensor
   DoReact("CAM","1","REC","rollback<1>");//ejecutar grabación en retroceso de la Primera cámara de video
}
```

VNS

El objeto **VNS** se corresponde con el objeto de sistema **Servicio de Notificación por Voz**.

El formato operador que describe las acciones del **VNS** es:

```
DoReact("VNS","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla muestra la lista de comandos y parámetros del objeto **VNS**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"SETUP" – configuración del servicio de notificación por voz.	card<>	Nombre del dispositivo de sonido. *Nota.El nombre de la tarjeta debe corresponderse con el nombre especificado en la configuración de la tarjeta de sonido del **Servicio de notificación por voz**.*
	level<>	Nivel de la señal. El valor del parámetro debe estar entre 0 y 15. El valor predeterminado es 8.
	channel<>	Conjunto de canales de sonido. Los valores de parámetro disponibles son: 0 – sin sonido; 1 – canal de reproducción izquierdo; 2 – canal de reproducción derecho; 3 – canales de reproducción izquierdo y derecho (ambos canales).
	flags<>	Indicadores.
	ip<>	Dirección IP del dispositivo de red.
	name<>	Nombre del objeto.
	pass<>	Contraseña.
	user<>	Nombre de usuario.
"PLAY" – play audio file.	file<>	Ruta completa y nombre del archivo de sonido. *Nota. Si solamente se especifica el nombre del archivo, se adoptará la ruta hacia éste registrada en «HKEY_LOCAL_MACHINE\SOFTWARE\ITV\Intellect» section (HKEY_LOCAL_MACHINE \Software\Wow6432Node\ITV\Intellectfor 64-bits system), en el valor del parámetro «InstalarRuta». En este parámetro, es posible reproducir varios archivos de sonido por medio de la operación «+».*
"ARM" – Activar sensor.	-	-
"DISARM" – Desactivar sensor.	-	-
"CONFIRM" – Confirmar alarma.	-	-
"SETUP" – Configuración del sensor.	chan<>	Número de salida (0 – 15).
	flags<>	Indicadores.
	name<>	Nombre del objeto.
	type<>	Tipo de objeto sensor (0 – al cerrarse, 1 – al abrirse).

La siguiente tabla muestra las propiedades del objeto **VNS**.

Propiedades del objetoVNS	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

Ejemplos de utilización de eventos y reacciones del objeto **Servicio de notificación por voz**:

1. Es necesario reproducir un archivo de sonido cuando la cámara de video inicie la grabación.

```
OnEvent("CAM",N,"REC")
{
   DoReact("VNS","1","PLAY","file<C:\Program Files\
Intellect\Wav\cam_alarm_"+N+".wav>");
}
```

2. Establecer nivelmenor del regulador de volumenpara el evento de la zona horaria especificada anteriormente y establecer el nivel medio en el regulador de volumen después de esta zona horaria.

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

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

> **Nota.**
> El objeto TIME_ZONE queda descrito de la siguiente manera (ver la sección TIME_ZONE).

SMS

El objeto **SMS** corresponde al objeto de sistema**Servicio de Mensajes Cortos (SMS)**.

El objeto **SMS** envía los eventos que aparecen en la tabla. El procedimiento se inicia cuando sucede el correspondiente evento.

El formato de procedimiento de eventos para elobjeto **Servicio de Mansajes Cortos (SMS)** es:

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

Descripción de los eventos del objeto **SMS**.

Evento	**Descripción**	**Comentario**
RECEIVE	Se ha recibido un mensaje	Utilice la clave de registroProcessFromSim si no se recibe el evento al recibir el mensaje (ver Guía de referencia de claves de registro). El texto del mensaje enviado se encuentra en el parámetro **mensaje <>**. El número de teléfono en formato +34XXXXXXXXXdesde el que se envió el mensaje está en el parámetro **teléfono**<>.

El formato operador que describe las acciones del servicio de mensajes cortos es:

```
DoReact("SMS","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla muestra la lista de comandos y parámetros del objeto **SMS**:

Comando – descripción del comando	**Parámetros**	**Descripción de los parámetros**
"SETUP" – settings of short message service.	device<>	Dispositivo SMS
	flags<>	Indicadores.
	message<>	Texto del mensaje.
	name<>	Nombre del objeto.
	phone<>	Número de teléfono.

Las propiedades del objeto **SMS** se muestran en la tabla.

Propiedades del objeto SMS	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

Ejemplos de utilización de eventos y reacciones del objeto **Servicio de Mensajes Cortos (SMS)**:

1. Es necesario enviar un SMS al número "+34644644644" al activarse una alarma en la primera cámara de video.

```
OnEvent("CAM","1","MD_START")
{
   DoReact("SMS","1","SETUP","phone<+34667667667>,message<Cámara 1, Alarma>");
}
```

2. Establecer dispositivo para el envío de mensajes y enviar mensaje al número "+34644644644" al activarse la alarma en el primer sensor.

```
OnEvent("GRAY","1","CONFIRM") //confirm alarm from sensor 1
{
   DoReact("SMS","1","SETUP","device<>",); //establecer dispositivo para envío de
mensajes

   DoReact("SMS","1","SETUP","phone<+34644644644>,message<Sensor 1, Alarma>");
//Enviar mensaje sobre la alarma en el sensor 1 al número de teléfono
}
```

3. Reproducir el archivo de audio c:\Windows\Media\Tada.wav al recibir un SMS utilizando el **Servicio de Mensajes de Correo 2**.

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

TELEMETRY

El objeto **TELEMETRY** se corresponde con el objeto de sistema **Controlador telemétrico**.

El objeto **TELEMETRY** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento en cuestión.

El formato de procedimiento de los eventos del objeto **Controlador telemétrico** es:

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

Descripción de eventos del objeto **TELEMETRY**.

Evento	Descripción	Comentario
LOCK	Bloqueado	Se recibe el evento después de aplicar el comando BLOQUEAR (ver el siguiente comando).
UNLOCK	Desbloqueado	Se recibe el evento después de aplicar el comando DESBLOQUEAR (ver el siguiente comando).

El formato operador que describe las acciones del objeto **TELEMETRY** es:

```
DoReact("TELEMETRY","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla muestra la lista de comandos y parámetros del objeto **TELEMETRY**:

Comando – descripción del comando	Parámetros	Descripción
"AUTOFOCUS_ON" – activar el enfoque automático	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"AUTOPAN_END_P" – especificar el punto de finalización del giro automático.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"AUTOPAN_START" – iniciar el giro automático.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"AUTOPAN_START_P" – especificar el punto de inicio de del giro automático.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"AUTOPAN_STOP" – detener el giro automático.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"CLEAR_PRESET" – borrar valor preestablecido seleccionado.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
	preset<>	Preestablecer.
"D2OFF" – desactivar ajustes dinámicos adicionales de las cámaras de video PTZ Panasonic, diseñadas para aumentar la calidad de la señal de video analógica.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"D2ON" – activar ajustes dinámicos adicionales de las cámaras de video PTZ Panasonic, diseñadas para aumentar la calidad de la señal de video analógica.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"DOWN" – rotar el objetivo de la cámara de video hacia abajo.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"FOCUS_IN" – acercar zoom.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"FOCUS_OUT" – alejar zoom.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"FOCUS_STOP" – detener acercar/alejar zoom en la imagen.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"GO_PRESET" – rotar la cámara de video en la posición especificada en el valor preestablecido.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
	preset<>	Preestablecer.
"HOME" – rotar la cámara de video a la posición inicial.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"IRIS_CLOSE" – cerrar el diafragma.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"IRIS_OPEN" – abrir el diafragma.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"IRIS_STOP" – detener el diafragma.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"LEFT" rotar el objetivo de la cámara de video hacia la izquierda.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).

"LEFT_DOWN" – rotar el objetivo de la cámara de video a izquierda y abajo.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"LEFT_UP" – rotar el objetivo de la cámara de video a izquierda y arriba..	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"PATROL_LEARN" – iniciar el procedimiento de programación de vigilancia registrando acciones de las cámaras de video.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"PATROL_PLAY" – iniciar vigilancia.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"PATROL_STOP" – detener vigilancia.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"RIGHT" – rotar el objetivo de la cámara de video a la derecha.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"RIGHT_DOWN" – rotar el objetivo de la cámara de video a la derecha y abajo.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"RIGHT_UP" – rotar el objetivo de la cámara de video a la derecha y arriba.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"SET_PRESET" – registrar la posición actual de la cámara de video en valores preestablecidos seleccionados.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
	preset<>	Preestablecer.
"STOP" – detener la rotación de la cámara de video.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"UP" – rotar el objetivo de la cámara de video hacia arriba.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta).
"SETUP" – configurardispotivoptz.	address<>	Dirección del dispositivo.
	cam<>	ID de la cámara a controlar.
	flags<>	Inidicador de funcionamiento del objeto (0 – ON, 1 - OFF).
	name<>	Nombre de objeto del dispotivoptz.
	speed<>	Velocidad.
"SEND_BUFFER" – enviar comando al puerto COM en formato hexadecimal.	buffer<>	Comandoenformato hexadecimal.
	parent_id<>	ID del objeto matriz del **Controlador telemétrico**. Parámetro necesario.
	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta). El valor del parámetro tiene que ser superior a 0.
LOCK - Bloquear. Cambiar el estado de la TELEMETRYa BLOQUEADO durante el tiempo especificado.	tel_prior<>	Prioridad (1 - baja, 2 – media, 3 – alta). El valor del parámetro tiene que ser superior a 0. No se permite ejecutar comandos con una prioridad menor que la especificada durante el tiempo de bloqueo.
	duration<>	Duración del bloqueo. Bloquear por la fuerza hasta la ejecución del comando DESBLOQUEAR si no se ha especificado el parámetro.
UNLOCK –desbloquear. Cambiar el estado de la TELEMETRYa DESBLOQUEADO durante el tiempo especificado.	-	-

La siguiente tabla muestra las propiedades del objeto **TELEMETRY**.

Propiedades del objetoTELEMETRY	**Descripción de las propiedades**
ID<>	ID del objeto.
PARENT_ID<>	ID del objeto matriz.

El objeto **TELEMETRY**puede encontrarse en los siguientes estados:

Estado del objetoTELEMETRY	**Descripción**

LOCKED - bloqueado	El control de la TELEMETRY está bloqueado con alguna prioridad. No se permite controlar la TELEMETRY con una prioridad mayor a la especificada al bloquearla (ver la tabla de abajo).
UNLOCKED - desbloqueado	Se permite controlar la TELEMETRY con cualquier prioridad.

Ejemplos de la utilización de eventos y reacciones del objeto **TELEMETRY**:

1. Establecer enfoque automático cuando la cámara de video está activada.

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

2. Rotar la cámara de video a la posición especificada en el primer valor preestablecido al activar el relé.

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

TELEMETRY_EXT

El objeto **TELEMETRY_EXT** corresponde al objeto de sistema **Teclado**.

El objeto **TELEMETRY_EXT** envía los eventos que se presentan en la tabla. El procedimiento se inicia cuando aparece el evento correspondiente.

El formato de procedimiento de los eventos para el objeto **Teclado** es:

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

Evento	Descripción del evento	Parámetro	Descripción del parámetro	Rango de los valores	
"KEY_PRESSED"	La tecla está pulsada	param0<>	Código de la tecla pulsada	Ver Guía de instalación y configuración de los componentes del sistema de seguridad.	
		device<>	El dispositivo en el que la clave está pulsada	0 – Teclado principal *AXIS T8312* , 1 - *AXIS T8313 keyboard*	
"KEY_RELEASED"	La tecla está liberada	param0<>	Código de la tecla liberada	0..21 para *AXIS T8312*. Para *BOSCH KBD-Digital, BOSCH KBD-Universal* y *Panasonic WV-CU950* ver Guía de instalación y configuración de los componentes del sistema de seguridad.	
		device<>	Dispositivo en el que la tecla está liberada	0 – Teclado principal *AXIS T8312*, 1 – Interruptor rotatorio *AXIS T8313*	
"MOVED"	Se ha cambiado la posición	param0<>	Valor de preferencia	Para el rotor del interruptor rotatorio JogDial -1.. 1; Para el rotor de Desplazamiento cuadro a cuadroShuttle -7..7 Para el teclado *Panasonic WV-CU950* JogDial -1.. 1; Shuttle -6..6	
		device<>	Tipo de mecanismo de control utilizado *AXIS T8313*	0 – rotor del interruptor rotatorio, 1 – rotor del desplazamiento cuadro a cuadro	

El formato operador para describir las acciones del objeto **TELEMETRY_EXT** es:

```
DoReact("TELEMETRY_EXT","_id_","_comando_" ["_parámetros_"]);
```

La siguiente tabla muestra la lista de comandos y parámetros para el objeto **TELEMETRY_EXT**:

Comando – descripción del comando	**Parámetros**	**Descripción**
"DRAW_FIGURE" – extraer la figura que se muestra en la pantalla de *BOSCH KBD-Digital o BOSCH KBD-Universal telemetry panel*	display<>	0x00 – pantalla principal, 0x01 – pantalla de estado
	x1<>	Iniciar coordinación con el eje X (de 0 a 127 en la pantalla principal, de 0 a 121 en la pantalla de estado)
	y1<>	Iniciar coordinación con el eje Y (de 0 a 239 en la pantalla principal, de 0 a 31 en la pantalla de estado)
	x2<>	Finalizar coordinación con el eje X (de 0 a 127 en la pantalla principal, de 0 a 121 en la pantalla de estado)
	y2<>	Finalizar coordinación con el eje Y (de 0 a 239 en la pantalla principal, de 0 a 31 en la pantalla de estado)
	is_fill<>	0 – Sin relleno, 1 – Relleno
	is_set_pixels<>	0 – eliminar la figura de la pantalla, 1 – extraer figura
	figure<>	0 – línea, 1 – rectángulo
"PRINT_TEXT" – imprimir el texto que se muestra en pantalla de *BOSCH KBD-Digital o BOSCH KBD-Universal telemetry panel*	display<>	0x00 – pantalla principal, 0x01 – pantalla de estado
	x<>	Coordinar el eje (de 0 a 127 en la pantalla principal, de 0 a 121 en la pantalla de estado)
	y<>	Coordinar el eje Y (de 0 a 239 en la pantalla principal, de 0 a 31 en la pantalla de estado)
	charset<>	Codificación: 0 – Latino 1 – Cirílico 2 – Centroeuropeo
	style<>	Estilo: 0 – Normal 1 – Semi-negrita
	text<>	Mensaje de texto
"PRINT_TEXT" – print text on display of *Panasonic WV-CU950 telemetry panel*	y<>	0 – mostrar el texto en la primera línea 1 – mostrar el texto en la segunda línea
	text<>	Texto introducido por línea, máximo 20 símbolos

	flickering<>	La línea consta de 6 símbolos que determinan los parámetros de intermitencia del texto: d1 d2 d3 d4 d5 d6 d1 determina el period de intermitencia: 0 – intermitencia desactivada 1 –periodo de 0.25 seg, se cambia el símbolo con un espacio en blanco 2 – periodo de 0.5 seg, se cambia el símbolo con un espacio en blanco 3 – periodo de 0.75 seg, se cambia el símbolo con un espacio en blanco. 4 – periodo de 1 seg, se cambia el símbolo con un espacio en blanco. 5 – periodo de 1.25 seg, se cambia el símbolo con un espacio en blanco 6 – periodo de 1.5 seg, se cambia el símbolo con un espacio en blanco 7 – periodo de 1.75 seg, se cambia el símbolo con un espacio en blanco 8 – periodo de 2 seg, se cambia el símbolo con un espacio en blanco d2: 1 – los símbolos del 1 al 4 son intermitentes, 0 – estos símbolos no son intermitentes. d3: 1 – los símbolos del 5 al 8 son intermitentes, 0 – estos símbolos no son intermitentes. d4: 1 – los símbolos del 9 al 12 son intermitentes, 0 – estos símbolos no son intermitentes. d5: 1 – los símbolos del 13 al 16 son intermitentes, 0 – estos símbolos no son intermitentes. d6: 1 – los símbolos del 17 al 20 son intermitentes, 0 – estos símbolos no son intermitentes.
" CLEAR_DISPLAY " – borrar contenido en pantalla de *BOSCH KBD-Digital o BOSCH KBD-Universal telemetry panel.* Reacción sin parámetros para el *Panasonic WV-CU950 telemetry panel*.	display<>	0x00 – pantalla principal, 0x01 – pantalla de estado
"RELE_ON" – encender la luz en el teclado *AXIS T8312* o *Panasonic WV-CU950panel*	rele<>	Código de la tecla con la luz,12..16 para *AXIS T8312*. Para Panasonic WV-CU950,ver Guía de instalación y configuración de los componentes del sistema de seguridad, sección Características de configuración y funcionamiento del panel de control Panasonic WV-CU950.
"RELE_OFF" – apagar la luz en el teclado *AXIS T8312* o *Panasonic WV-CU950 panel*	rele<>	Código de la tecla con la luz, 12..16
"RESET" – reiniciar el *Panasonic WV-CU950 panel*	type<>	0 – reinicio inmediato 1 – reinicio tras 100 ms. 2– reinicio tras 200 ms. 3– reinicio tras 500 ms. 4– reinicio tras 1 s.

"SET_ALARM" – establecer el tipo de señal de alarma del *Panasonic WV-CU950 panel*	audio_alarm<>	0 – sin sonido 1 – señal de alarma única simple 2 – señal de alarma doble simple 3 – señal de alarma triple simple 4 – señal de alarma única de duración de 0.1 seg. 5 – señal de alarma única de duración de 0.2 seg. 6 - señal de alarma única de duración de 0.3 seg. 7 - señal de alarma única de duración de 1 seg. 8 – Tono único simple 9 – Tono doble simple A– Tono triple simple B– Señal única de duración de 0.1 seg C– Señal única de duración de 0.2 seg D– Señal única de duración de 0.3 seg E– Señal única de duración de 1 seg F – Señal de alarma

Ejemplo de utilización de eventos y reacciones del objeto **TELEMETRY_EXT**:

1. Encender la luz y activar la cámara 2 al pulsar la tecla 15 en el teclado AXIS T8312.

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