Security Systems

Nuremberg
2017-03-01

From ST-VS/MKI	Our Reference	Tel	Fax

Technical information

Topic **How to receive IVA events via RCP+ ALARM_OVERVIEW**

1 Introduction (IVA/EVA Release letter)

Intelligent Video Analytics and Essential Video Analytics are software algorithms that detects behavior of objects within an environment monitored by a video camera and generates alarm events that can be processed further in a CCTV system. They make it possible to capture and evaluate directional movement of objects, apply configured filter rules and to combine these rules, thereby largely preventing false alarms. Both algorithms adapts automatically to changing environmental conditions and are therefore non-sensitive to perturbing influences such as rain and small tree movements.

Intelligent Video Analytics is more robust than Essential Analytics and can cope with heavy rain and snow falls, strong wind moving vegetation or shaking the camera, and water surfaces and fountains in the background.

Both algorithms provide easy to use, intuitive configuration via graphical user interface as a part of an advanced wizard structure in the Task Manager. Improved with an intuitive configuration option it is possible to provide the complete property information (object type, size, speed, aspect ratio, direction, color) for an object just by clicking it in the live scene.

Intelligent Video Analytics has the auxiliary function “Flow”, which supports optical flow to detect the direction of objects independent from the structure of the background and the size of the objects. It also has the ability to detect frontal faces and upload the best face snapshot to an ftp server or Dropbox account.

The metadata generated by Intelligent Video Analytics and Essential Video Analytics, transmitted to live video or to storage, is used to display overlay graphics and allow retrospective forensic search.

2 Browser Compatibility

The configuration button in the Alarm-VCA menu works with Browser which allows ActiveX Container for NPAPI like Internet Explorer 11, Firefox 48 or Safari 5. Browsers which deactivated ActiveX by default like Google Chrome or Microsoft Edge don't show the full settings / surveillance tasks for the VCA configuration. Furthermore only one client is allowed at the same time to configure the IVA tasks. Any additional clients get the message 'Configuration in progress. No further configuration possible'.

3 General usage of the ALARM_OVERVIEW (0x0c38)

With FW 6.30 Bosch IP cameras got an update on the alarm retrieval mechanism. Previously, analytics alarms, audio alarms, virtual alarms, I/Os … had dedicated message subscriptions, which ended with a huge list of possible subscriptions. With FW 6.30 a new, unified, command / subscription was introduced.

The CONF_ALARM_OVERVIEW (0c0c38) command combines all available events in one single subscription.

	Tag code	NumDes	Message	SNMP Support
	0x0c38	video line (starts with 1)	yes	no
	Datatype	**Access Level**	**Description**	
Read	p_octet	access_right_noprot	see detailed description below	
Write	*p_octet*	*access_right_service*	*not supported*	

The command can be used as a 'regular' RCP+ command, but it can also be used as a message subscription with slightly different answers to be expected. The command gives an overview on the current setup and a message is sent each time, one of the enclosed entities is changed.

The general payload format of the command is as follows:

Flags	Reserved	Number of alarm entries
1 Byte	1 Byte	2 Byte

Alarm Entry n Byte
…
Alarm Entry n Byte

```
Flags
   Values: 0x80 – All entries included (read out)
```

The Flag can be used to distinguish between a ‘status overview reply’ or an event message. The status overview contains all entries and hence is signaled with a value of **0x80** (see sections below for further information).

One “Alarm Entry” can be interpreted as depicted below:

<table>
<tr><td colspan="2"><b>Entry ID</b>
2 Byte</td><td colspan="2"><b>Entry Length</b>
2 Byte</td></tr>
<tr><td><b>Flags</b>
1 Byte</td><td><b>Reserved</b>
1 Byte</td><td><b>Alarm Source</b>
1 Byte</td><td><b>Alarm Type</b>
1 Byte</td></tr>
<tr><td colspan="4"><b>Alarm Name</b>
n Byte</td></tr>
</table>

```
Entry ID: Unique ID for each entry.
   May be reassigned to entry if      entry is deleted.
Entry Length: Length of the entry payload, exclusive header.

Flags[1]
 Values:
 0x80   Add Flag            Alarm Entry has been registered
 0x40   Delete Flag         Alarm Entry has been unregistered
```

[1] Multiple flags may be combined with the binary OR operator

```
  0x20  State Flag        Alarm Entry is a state
  0x10  State Set Flag    Alarm Entry State is set (states only)

Alarm Source
 Values:
 0   Unknown
 2   Relais
 3   Digital Input
 4   Audio
 5   Virtual Input
 6   Tamper
 7   Motion
 8   Flow
 9   Intelligent Video Analytics
 10  Fire

Alarm Type
 Values:
 0   Unknown
 1   VCA
 2   Relais
 3   Digital Input
 4   Audio
 5   Virtual Input
 16  Default Task
 17  Global Change
 18  Signal too bright
 19  Signal too dark
 23  Reference Image Check Failed
 24  Invalid Configuration
 25  Flame Detected
 26  Smoke Detected
 32  Object in field
 33  Crossing line
 34  Loitering
 35  Condition change
 36  Following route
 37  Tampering
 38  Removed object
 39  Idle object
 40  Entering field
 41  Leaving field
 42  Similarity search
 43  Crowd detection
 44  Flow in field
 45  Counter flow in field
 46  Motion in field
```

3.1 As command

Using CONF_ALARM_OVERVIEW as a 'regular' command gives an overview of the current 'Alarm State' of the device. The numeric descriptor can be used to address a single line on an encoder.

The device's answer to the command results in a list of 'available events' on the device and for some events the current state is signaled. The device basically distinguishes between two types of events. Singular events and those that have a state (like I/Os, the Tamper events, ...).

3.1.1 Payload

Flags

The events that have a state are marked with the **'State Flag' (0x20**). Some of these events may be in 'high state' while the command is received and a message is generated. The high state of such an 'event-alarm' is signaled with the **'State Set Flag' (0x10)** and will result in a flag value of **'0x30'**.

The 'singular events' (like most of the IVA events) are signaled with a flag value of **'0x00'**, which simply means that they are available on the device and don't have a state.

Alarm source

The alarm source describes the initiator of the alarm. That can be a hardware component of the device or a software component. This entry can be used for a quick evaluation of the relevance of the alarm.

Alarm type

Specifies the alarm type, depending on the Alarm source. For some sources the type is simply the same as the source, but especially for analytics events this is valuable information.

3.1.2 Examples

A typical answer on an ALARM_OVERVIEW request may look like the following lines (after parsing the payload):

```
CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x80

EntryId 1 Flags: 0x20    - - -        |           |  STATE  |
                 -- Source: Tamper -- Type: Default Task
                 ---> 'Default Task'

EntryId 2 Flags: 0x20    - - -        |           |  STATE  |
                 -- Source: Tamper -- Type: Global Change
                 ---> 'Global Change'

                            8<  snip 8<

EntryId 9 Flags: 0x20    - - -        |           |  STATE  |
                 -- Source: Fire -- Type: Flame Detected
                 ---> 'Flame Detected'

EntryId 10 Flags: 0x20   - - -        |           |  STATE  |
                 -- Source: Fire -- Type: Smoke Detected
                 ---> 'Smoke Detected'

EntryId 11 Flags: 0x20   - - -        |           |  STATE  |
                 -- Source: Digital Input -- Type: Digital Input
                 ---> '0.0.0.0 - Input - 01'

EntryId 12 Flags: 0x20   - - -        |           |  STATE  |
                 -- Source: Digital Input -- Type: Digital Input
                 ---> '0.0.0.0 - Input - 02'

                             8<  snip 8<

EntryId 17 Flags: 0x20   - - -        |           |  STATE  |
                 -- Source: Virtual Input -- Type: Virtual Input
                 ---> 'Virtual Alarm 3'

EntryId 18 Flags: 0x0    - - -        |           |         |
                 -- Source: Intelligent Video Analysis -- Type:
                            Object in field
                 ---> 'Detect any object'
```

Entries 1 – 17 are Events with a state and none of the events is currently at high state (Flag = 0x20).

Entry 18 is a 'Video Analytics Event' without a state and of the type "Object in field". The name of the event is 'Detect any object'.

3.2 As message

Using CONF_ALARM_OVERVIEW as a 'message-subscription' initially responds in the same way as the command (described above) to represent the current state, but with subsequent message subscriptions only state changes or singular events are transmitted. The Event Flag (either **0x80** or **0x00**; see also Page 3) can be used to distinguish between an update message or an overview message.

3.2.1 Payload

Flags

As messages are also generated in case the previously announced information is changed, additional flags are needed, to be able to represent this changes. These changes are usually only relevant for the analytics configuration. A message is sent in case a analytic rule is added / enable or deleted / disable. These changes are represented by the flag values: **0x80** (event added) and **0x40** (event deleted)

Alarm source

See description above.

Alarm type

See description above.

3.2.2 Example

A typical message may look like the following lines (after parsing the payload):

Removing one analytic rule:

```
CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x0

EntryId 19 Flags: 0x40   - - -        |  DELETED  |         |
                 -- Source: Intelligent Video Analysis -- Type:
                            Crossing line
                 ---> 'Crossing line 2'
```

Adding an analytic rule:

```

CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x0

EntryId 18 Flags: 0x80   - - - ADDED  |           |         |
                 -- Source: Intelligent Video Analysis -- Type:
                            Crossing line
                 ---> 'Crossing line 2'
```

Alarm event triggered by an analytic rule:

```
CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x0

EntryId 18 Flags: 0x0    - - -        |           |         |
                 -- Source: Intelligent Video Analysis -- Type:
```

```
                            Crossing line
                 ---> 'Crossing line 2'
```

Relais set to high

```
CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x0

EntryId 13 Flags: 0x30   - - -      |         |  STATE  |  STATE SET
                 -- Source: Relais -- Type: Relais
                 ---> '0.0.0.0 - Output - 01'
```

Releais set to low

```
CONF_ALARM_OVERVIEW - printout
---------------------------------------
Flags: 0x0

EntryId 13 Flags: 0x20   - - -      |          |  STATE  |
                 -- Source: Relais -- Type: Relais
                 ---> '0.0.0.0 - Output - 01'
```

4 Detailed sample with payload parsing

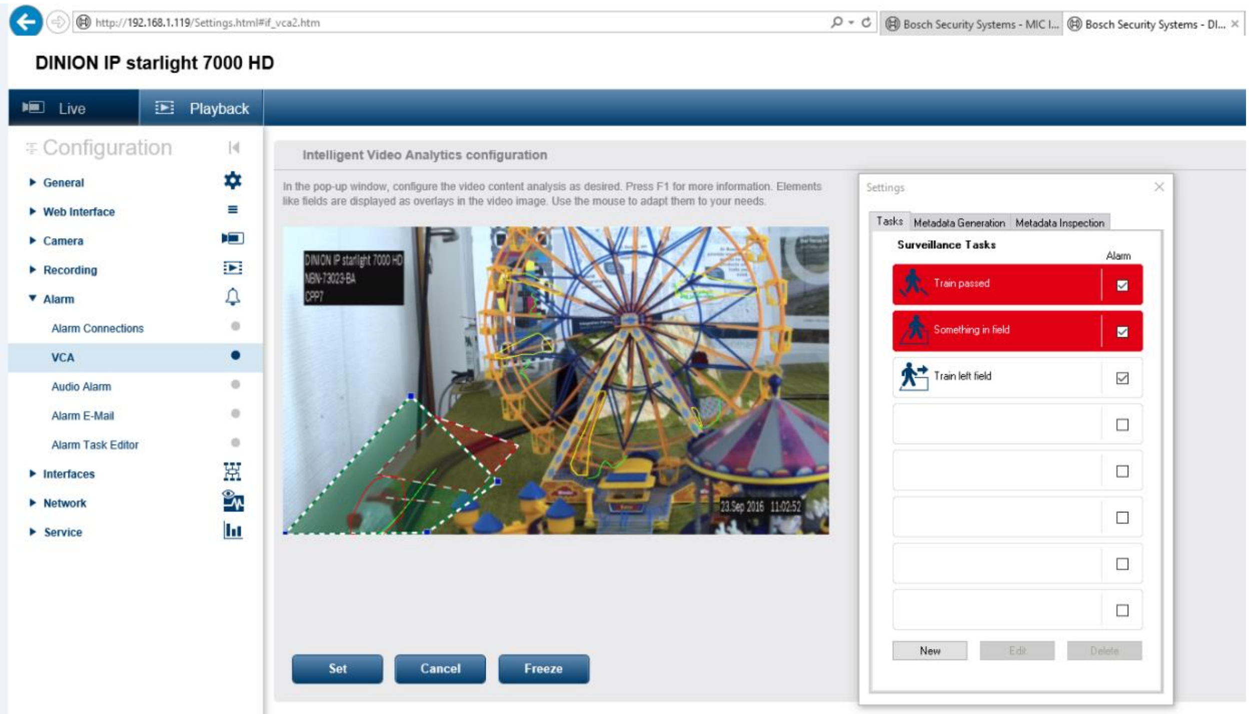
This section is supposed to demonstrate the payload parsing of an alarm message. The Intelligent Video Analytics events that are used in this sample are “Crossing line” and “Object in field”.

4.1 Crossing line and Object in field

Nuremberg
2017-03-01

From	Our Reference	Tel	Fax
ST-VS/MKI			

Topic **How to receive IVA events via RCP+ ALARM_OVERVIEW**



DINION IP starlight 7000 HD

Intelligent Video Analytics configuration

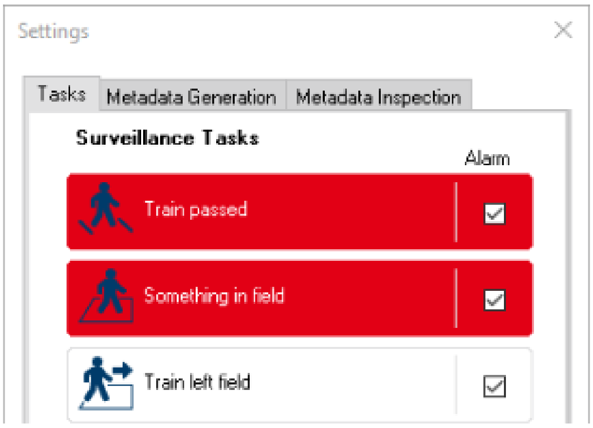
In the pop-up window, configure the video content analysis as desired. Press F1 for more information. Elements like fields are displayed as overlays in the video image. Use the mouse to adapt them to your needs.

Settings

Tasks | Metadata Generation | Metadata Inspection

Surveillance Tasks

	Alarm
Train passed	☑
Something in field	☑
Train left field	☑
	☐
	☐
	☐
	☐
	☐



Settings

Tasks | Metadata Generation | Metadata Inspection

Surveillance Tasks

	Alarm
Train passed	☑
Something in field	☑
Train left field	☑

The analytics tasks are configured via the web browser:

1. Type: “Crossing line” Name: ‘Train passed’
2. Type: “Object in field” Name: ‘Something in field’
3. Type: “Leaving field” Name: ‘Train left field’

In the example two rules are matching and trigger a message. The 3rd rule doesn't and won't trigger a message

The following RCP+ command get the even overview via the CGI interface:

http://192.168.1.119/rcp.xml?command=0x0c38&type=P_OCTET&direction=READ&num=1

The following RCP+ message subscribes to the message notification:

http://192.168.1.119/rcp.xml?message=0x0c38&type=P_OCTET&direction=READ&num=1&collectms=5000

The response payload is to be interpreted as described above. Triggered rules 1 and 2 create the following message:

```
<?xml version="1.0"?>
<message_list>

  <msg>
    <command>0x0c38</command>
    <num>0</num>
    <cltid>0x0658</cltid>
    <hex>0x000000010013002c000009200053006f006d0065007
    400680069006e006700200069006e002000660069006500 6c0064</hex>
  </msg>

  <msg>
    <command>0x0c38</command>
    <num>0</num>
    <cltid>0x0658</cltid>
    <hex>0x000000010012002000000921005400720061006900
    6e00200070006100730073006500 64</hex>
  </msg>

</message_list>
```

Message 1 payload

```
<hex> 0x00000001 0013002c 00000920

      0053006f06d0065007400680069006e0

      06700200069006e00200066006900650

      06c0064

</hex>

0x00000001 -> Event flags (1b): 0x00

              Reserved (1b): 0x00

              Number of entries: 0x0001

0x0013002c -> Entry ID (2b): 0x0013 (19)

              length (2b): 0x002c (44) byte

0x00000920 -> Flags (1b): 0x00

              Reserved (1b): 0x00

              Alarm source (1b): 0x09 (IVA)

              Alarm type (1): 0x20 (32, Object in field)

0x53 0x6f 0x6d 0x65 0x74 0x68 0x69 0x6e 0x67 0x2 0x069 0x6e 0x2 0x066
0x69 0x65 0x6c 0x64 -> „Something in field"
```

Message 2 payload

```
<hex> 0x00000001 00120020 00000921

      0054007200610069006e002000700061
      0073007300650064

</hex>

0x00000001 -> Event flags (1b): 0x00

              Reserved (1b): 0x00

              Number of entries: 0x0001

0x00120020 -> Entry ID (2b): 0x0012 (18)

              length (2b): 0x0020 (32) byte

0x00000921 -> Flags (1b): 0x00

              Reserved (1b): 0x00

              Alarm source (1b): 0x09 (IVA)

              Alarm type (1b): 0x21 (33, Crossing line)

0x54 0x72 0x61 0x69 0x6e 0x02 0x07 0x061 0x73 0x73 0x65 0x64 -> "Train
passed"
```

In case no events happen you receive an empty message list:

```
<?xml version="1.0"?>
<message_list> </message_list>
```