

Alarm Task Script Language

Version 4.1



BOSCH

Table of Contents

1	Introduction	1
2	Definitions	1
2.1	Actions	1
2.2	Events and States	1
2.3	Alarm Task Engine	1
3	System Integration	2
4	Configuration	2
5	Syntax	4
5.1	Action Types	4
5.1.1	Send Alarm E-Mail	4
5.1.2	JPEG Posting	5
5.1.3	Recording	6
5.1.4	Connection	7
5.1.5	Connection List	9
5.1.6	RCP Command	10
5.1.7	VCA Configuration	11
5.1.8	Operation Mode	11
5.1.9	Timer	12
5.1.10	Stopping of an Action	13
5.2	State Types	14
5.2.1	I/O-States	14
5.2.2	Tamper States	15
5.2.3	Action States	15
5.3	Conditions	17
5.3.1	Boolean Composition of Conditions	17
5.3.2	State Condition	17
5.3.3	State Operation Mode	17
5.3.4	Action Condition	18
5.4	Comments	19
6	Starting the Alarm Task Engine	19
7	Glossary	20

1 Introduction

The Alarm Task Engine is an extension of the alarm I/O (Input/Output) matrix. It supports the definition of actions, action timers, temporary states and I/O-states. Furthermore the Alarm Task Engine supports conditions when actions shall be triggered or states shall be set.

2 Definitions

2.1 Actions

Each device provides various actions. Actions could be sending of an alarm e-mail, JPEG posting (Joint Photographic Expert Group), sending of RCP commands (Remote Control Protocol) or connecting to another device and so on. With the script language it is possible to define and to configure the different actions and the according parameters like IP address (Internet Protocol) or password and more.

2.2 Events and States

The script language provides the Boolean evaluation of various kinds of states. Each device has a different number of I/O-states. These could be relays, alarm inputs, connection state and so on. Such states are either enabled or disabled. The numbers of I/O-states depend on the particular device. Furthermore, there are states which depend on the particular defined actions. With these states it is possible to query whether the execution of the action was successful or not. Another possibility is to query whether a particular connection is even active or not to a decoder or encoder. All I/O-states and actions states are distinguished between:

- Readable (R)
- Readable/Writable (R/W)
- Configurable (C)

R-state means, you can just query the current value of them. If you use R/W-states, you can change the value of it. And C-state can be used for setting to different kind of operation mode like bistable, monostable or periodic.

An event is always created when the I/O-state is changing. Consequently, the event has the temporal information about which state has changed from disabled to enabled or vice versa.

2.3 Alarm Task Engine

The Alarm Task Engine parses all events of the I/O-Manager module and evaluates changes of the I/O-states. The Alarm Task Engine receives the events from the I/O-Manager and triggers the according defined actions.

3 System Integration

Figure 3.1 shows the architecture which is relevant for the Alarm Task Engine. The I/O-Manager receives all types of events. Such events could be the changing of an alarm or an activated video input and so on. Furthermore, if the Alarm Task Engine is running, it can create events too. All these created events are fed into the I/O-Manager. Consequently, the I/O-Manager knows all I/O-states and evaluates whether the state has been changed or not. Hence, it can be guaranteed that the Alarm Task Engine is only running if a state has been changed.

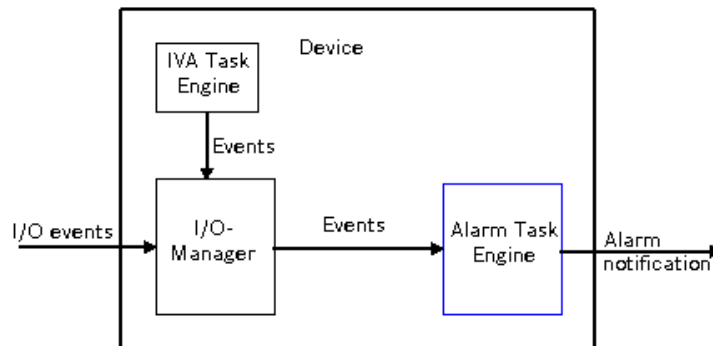


Figure 3.1 Alarm Task Engine Architecture

4 Configuration

The Alarm Task Engine is configured by a script. There is the option to create scripts automatically or you can input the script directly. The script can be configured automatically under the browser page `\Settings.html`:

- Alarm > Alarm Connections
- Alarm > Alarm E-Mail
- Network > JPEG Posting
- Interfaces > Relay

If you use these options, the script is created and sent to the device automatically.

You can find the editor in the browser under the page `\Settings.html`:

- Alarm > Alarm Task Editor

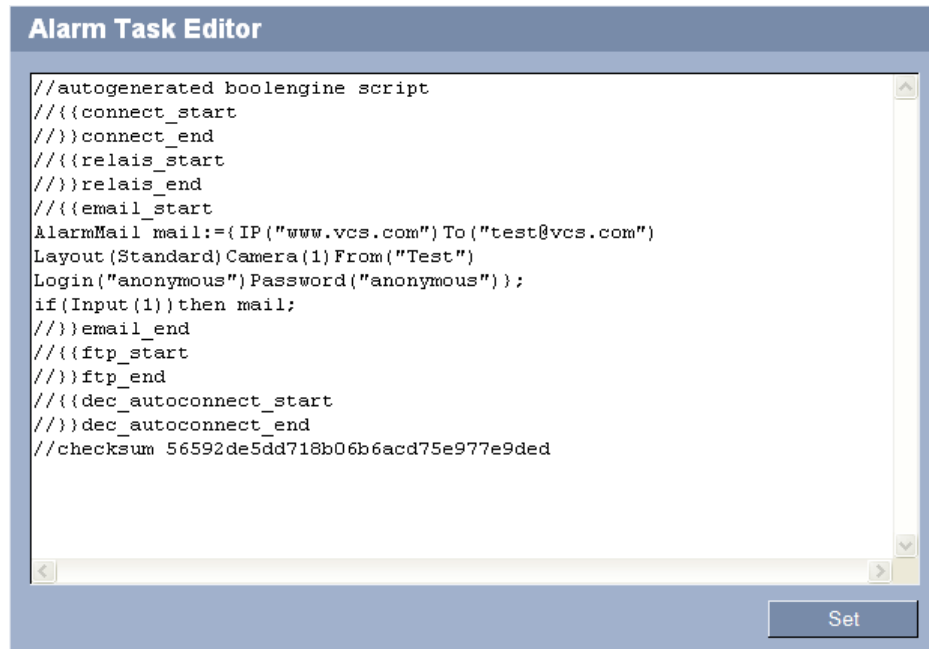
Figure 4.1 shows the user defined settings for sending an alarm e-mail.

Alarm E-Mail

Send alarm e-mail	<input type="text" value="On"/>
Mail server IP address	<input type="text" value="www.vcs.com"/>
SMTP user name	<input type="text" value="anonymous"/>
SMTP password	<input type="password" value="••••••••"/>
Layout	<input type="text" value="Standard (with JPEG)"/>
Attach JPEG from camera	<input checked="" type="checkbox"/> 1
Destination address	<input type="text" value="test@vcs.com"/>
Sender name	<input type="text" value="Test"/>
Test e-mail	<input type="button" value="Send now"/> <input type="button" value="Set"/>

Figure 4.1 Alarm E-Mail Dialog Box

And *Figure 4.2* displays the automatically created script. You can see in the script an alarm mail will be sent to the entered mail server IP (Internet Protocol) address, if an alarm input is activated.

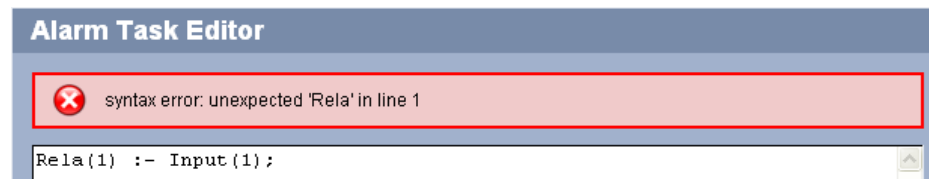


```
//autogenerated boolengine script
//{{connect_start
//}}connect_end
//{{relais_start
//}}relais_end
//{{email_start
AlarmMail mail:={ IP ("www.vcs.com") To ("test@vcs.com")
Layout (Standard) Camera (1) From ("Test")
Login ("anonymous") Password ("anonymous") };
if (Input (1)) then mail;
//}}email_end
//{{ftp_start
//}}ftp_end
//{{dec_autoconnect_start
//}}dec_autoconnect_end
//checksum 56592de5dd718b06b6acd75e977e9ded
```

Figure 4.2 Alarm Task Engine Script

It is also possible to edit the automatically created actions and conditions by hand. But after that, you can not configure the current script in the browser again without losing the current script – because the new script will replace the current script.

If you enter the script by hand and the script has at least one syntax error the separate errors and the according line will be listed in the message box above the script editor as shown in *Figure 4.3*. A syntax error could occur when you have entered a password or an URL (Uniform Resource Locator) which is too long. Another error could be when you have forgotten a bracket or a semicolon and so on. Further, it could be displayed warnings or if the script does not have a syntax error the message: "Script successfully parsed." appears.



```
Rela(1) :- Input(1);
```

Figure 4.3 Alarm Task Engine Error Message



NOTICE!

It must be pointed out that the maximum script size is about 4 KB. It depends on the zipping result. If the script gets too big, the script can not be saved on the device and a write error occurs.

5 Syntax

5.1 Action Types

The script language supports the configuration of various action types. Each action is configured by standard and optional parameters. The general syntax of actions is:

```
<action name> <identifier>:={<standard parameters>
[<optional parameters>]};
```

It is differentiated in asynchronous and synchronous actions. Asynchronous actions can take several seconds before they are finished or started. And in contrast the synchronous actions will execute immediately. The following list shows the <action name> and whether the action is asynchronous or synchronous:

Asynchronous:

- AlarmMail
- JPEGPosting
- Recording
- Connection
- ConnectionList
- RcpCommand
- VCAConfiguration

Synchronous:

- OperationMode
- Timer

An <identifier> begins with a lower case letter that can be followed by an upper case, a lower case letter, a digit or an underscore e.g.: alarmMail_1 or con_23. The maximum length of an identifier is 31 characters. Each action has at least one parameter. All parameters have the same syntax, which is as follows:

```
<parameter name>(<parameter>)
```

The order of the standard parameters should be entered correctly. In contrast to the standard parameters, the optional parameters have no fixed order.

5.1.1 Send Alarm E-Mail

Alarms can be documented by e-mail. This allows notification of clients which do not have a video receiver. So, you can define an action that automatically sends an e-mail to a previously defined e-mail address.

Standard parameters:

- IP("192.168.0.1")
Specify the IP or URL address of an e-mail server that operates on the SMTP standard (Simple Mail Transfer Protocol) here. The maximum URL length is 127 characters.
- To("test@vcs.com")
Enter the e-mail address for alarm e-mails here. The maximum e-mail length is 127 characters.

Optional parameters:

- `Layout (<format>)`
You can select the data format of the alarm message. The `<format>` can be:
 - `Standard`: attached with one or more JPEG image files.
 - `SMS (Short Message Service)`: E-mail is sent in SMS format to an e-mail-to-SMS gateway (for example to send an alarm to a cell phone) without an image attachment. This value is the default format.
- `From ("test@vcs.com")`
Enter the sender e-mail address here. The default address is `videosever`.
- `Subject ("Alarm Mail")`
Enter the e-mail subject here. The default subject is `alarm`.
- `Password ("anonymous")`
If the mail server is password protected, you can enter the password here.
- `Login ("anonymous")`
Enter the login for the e-mail server here.
- `Camera (1,2,3,...)`
For selecting image files you must list the particular cameras. If you select all cameras, you can also use the word `All`. The default value is empty. That means, no image file will be sent.
- `Name ("Action Name")`
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:

```
AlarmMail mail:={IP("192.168.0.1")To("to@vcs.com")
Layout(Standard)From("from@vcs.com")
Subject("test")Password("anonymous")
Login("anonymous")Camera(1,3)Name("alarm email");}
```

5.1.2**JPEG Posting**

You can define an action for individual posting one or more JPEG images on an FTP server (File Transfer Protocol).

**NOTICE!**

If you change the script, all open FTP connections are closed.

Standard parameters:

- `IP ("192.168.0.1")`
Enter the IP or URL address of the FTP server on which you wish to save the JPEG images here. The maximum URL length is 127 characters.
- `Login ("anonymous")`
Enter the login of the FTP server here; the maximum length is 31 characters.
- `Password ("anonymous")`
Enter the password of the FTP server here; the maximum length is 31 characters.

Optional parameters:

- `Format(<formats>)`
Select the resolution of the JPEG images to have. You have the selection of three formats:
 - `Small`: 176×144/120 pixels (QCIF)
 - `Medium`: 352×288/240 pixels (CIF). This value is the default.
 - `Large`: 704×567/480 pixels (4CIF)
- `Path("root/")`
Enter the path on the FTP server here. The default path is an empty string.
- `FileName("jpegPosting")`
You can specify the file name. The default name is `snap`. The files which are archived on the FTP server will always have an additional suffix. The maximum length of the file name is 31 characters minus the suffix length and the extension `.jpg`. If the name is too long it is cut off at the maximum length.
- `Suffix(<suffix>)`
You can select how file names will be created for the individual images which are transmitted. The `<suffix>` can be:
 - `Overwrite`: The same file name is always used. Some existing file will be overwritten with the current file name. This suffix is also the default value.
 - `Increment`: A number from 000 to 255 is added to the file name, for instance `snap_001_c1.jpg`, and automatically incremented by 1. When it reaches 255 it will start again from 000. If you modify the script, the suffix would also start from 000. If the server is not available, the number will be increased nonetheless. This means that the generated file names with this numbers become lost.
 - `Date`: The date and time are automatically added to the file name. When setting this parameter, ensure that the unit's date and time are always correctly set. Example: the file `snap_c1_011005_114530.jpg` of camera 1 was stored on October 1, 2005 at 11:45:30 a.m.
- `Camera(1,2,...)`
For selecting the JPEG files you must list the particular cameras. If you select all cameras, you can also use the word `All`. The default camera is 1.
- `Name("Action Name")`
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:

```
JpegPosting posting:={IP("192.168.0.1")Login("anonymous")
Password("anonymous")Format(Small)Path("test")FileName("alarm")
Suffix(Increment)Camera(All)Name("Jpeg posting")};
```

5.1.3**Recording**

It is possible to define an action that sends a command for starting and stopping the recording. The various possibilities of stopping the recording are explained in detail in *Section 5.1.10 Stopping of an Action*, page 13. This action has only one standard parameter. The recording will be able to start if there is a storage medium and the recording scheduler is enabled.

**NOTICE!**

If you change the script, the activated recording will not be stopped because of the possibility the recording is started from another client. If you want to stop the recording, you have to do it manually.

Standard parameter:

- Camera (1, 2, ...)

You can list all cameras for which the recording should be started or stopped. If you want to select all cameras, you can use the word `All`.

Example:

```
Recording recording:={Camera (1, 2, 3) };
```

5.1.4**Connection**

You can define a video and an audio connection from an encoder to a decoder or reversely. The device, where you enter the script is the local and the other one the remote. It is possible to start and stop a connection. Stopping a connection – more precisely disconnecting an active connection – is described in *Section 5.1.10 Stopping of an Action*, page 13.

**NOTICE!**

If you change the script, all open connections will be closed.

Standard parameter:

- IP ("www.bosch.com")

Enter the IP or URL address of the device you would like to connect here. The maximum URL length is 127 characters.

**NOTICE!**

If you use a decoder the optional parameter `LocalDirection` (see below) must be set.

Optional parameters:

- Protocol (<protocol>)

Select one of these protocols:

- UDP (User Datagram Protocol), the default value. It is recommended to use the UDP protocol.
- TCP (Transmission Control Protocol)

- LocalLine (<line>)

Enter the local line as integer that should be connected to a remote station. The default line is 0. That means, the first line, where a video is activated, will be chosen.

- LocalCoder (<coder>)

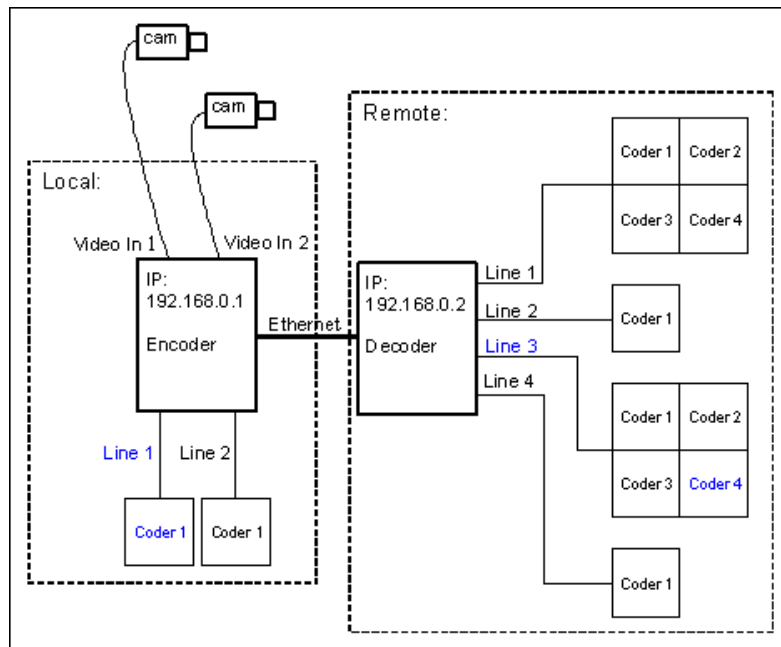
A coder is a monitor or view where you can see the video. The default coder is 0. That means, the first free monitor, where a video is displayed, will be chosen.

- LocalDirection (<direction>)

Choose from the following options:

- Out (outgoing): Use this option if you are using an encoder. This option is the default value.
- In (incoming): Use this option if you are using a decoder.
- Bi (bidirectional): For devices which support both outgoing and incoming.

- RemoteLine (<line>)
Enter the remote line as integer where the connection should be established. The default line is 0. That means, the first free line of the remote station will be chosen.
- RemoteCoder (<coder>)
A coder is a monitor or view where you can see the video. The default coder is 0. That means, the first free monitor of the remote station will be chosen.
- RemotePort (<port>)
Depending on the network configuration, you can set a browser port:
 - HTTP (Hypertext Transfer Protocol): This value designates port number 80 (the default value).
 - HTTPS (Hypertext Transfer Protocol Secure): This value designates port number 443 and enables the encryption mode automatically.
 - Enter a valid port number manually. If you use port number 443 the SSL (Secure Socket Layer) encryption mode will be set to `true`, automatically.
- SSL (<boolean>)
You can set the encryption mode <boolean> to `true` or `false`. If you want to use HTTPS connection you must enable the SSL mode. The SSL mode is by default `false`.
- Audio (<boolean>)
Enter for <boolean> `true` to enable and `false` to disable the audio connection; `false` is also the default value.
- Password ("anonymous")
If the remote station is password protected, enter the password here. The default password is an empty string and the maximum length is 31 characters.
- VideoCoding (<coding>)
Enter one of the coding types:
 - MPEG2 (Moving Picture Experts Group)
 - MPEG4
 - H264
 - All: The best match will be attempted for connection. This is the default value.
- VideoSubstitute (<boolean>)
If a connection already exists, you can disconnect the old one with the value `true` and substitute it with the new one. The default value is `false`.
- Name ("Action Name")
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:**Figure 5.1** Encoder/Decoder Connections

In *Figure 5.1* you can see the connection from an encoder to a hardware decoder. In this figure the encoder is the local station and the decoder the remote. A decoder device has one or four views per line. They are named single or quad view. You can set the coder per line where you want to see the video. In the following practical example, the video from the encoder, Line 1 and Coder 1, should be established to the decoder, Line 3 and Coder 4:

```
Connection con_1:={IP("192.168.0.2")LocalLine(1)LocalCoder(1)RemoteLine(3)
RemoteCoder(4)};
```

5.1.5**Connection List**

With this action it is possible to define the order how connections should be activated until a connection has been established. But it is also possible to stop this action. That is explained in *Section 5.1.10 Stopping of an Action*, page 13.

Standard parameter:

- `Connection(con_1, con_2, ...)`
List the connections here. Before you enter the connections you must define all of these (see *Section 5.1.4 Connection*, page 7). The action will try to connect the first one after about 10 seconds, then the next one until a connection has been established.

Optional parameter:

- `AutoConnect(<boolean>)`
You can enable the `AutoConnect` property with `true` or disable with `false` (default). If the property is activated, the connection list is restarted after one second delay until a connection has been established. On the other hand, if `AutoConnect` is disabled, the connection list will not restart again.
- `Name("Action Name")`
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:

```

Connection con_1:={IP("192.168.0.1")};
Connection con_2:={IP("192.168.0.2")};
ConnectionList autoConnect:={Connection(con_1,con_2)AutoConnect(true)
Name("Connection List")};

```

5.1.6**RCP Command**

With this action you can send RCP commands to the device itself (local host) or to another device.

Standard parameter:

- Command(<rcp command>)

The <rcp command> has the same structure as the CGI command (Common Gateway Interface). The according command can be created on the debug page: \dbg_rcpcmd. So you can copy the created CGI command and enter it here.

Optional parameter:

- SSL(<boolean>)

You can with `true` enable or with `false` disable the encryption mode for sending the RCP command.
- Port(<port>)

Enter the <port> for sending the RCP command here:

 - HTTP: This value designates port number 80 (the default value).
 - HTTPS: This value designates port number 443 and enables the encryption mode automatically.
 - Enter a valid port number manually. If you use port number 443 the encryption mode is enabled automatically.
- Password("anonymous")

If the remote device is password protected, enter the password here. The default password is an empty string and the maximum length is 31 characters.
- UserName(<username>)

Enter the corresponding user name for the given password here. The user name defines the authorization level. Possible levels for <username> are:

 - Service: This is the highest authorization level (default value). After entering the correct password, this user name allows you to use all the functions of the device and change all configuration settings.
 - User: This is the middle authorization level. With this authorization level, you can operate the device, but you cannot change the configuration.
 - Live: This is the lowest authorization level. With this authorization level, you can view live video, but you cannot operate the device or change the configuration.
- IP("192.168.0.1")

Enter the IP or URL address here. The default IP address is 127.0.0.1 (local host). The maximum URL length is 127 characters.
- Name("Action Name")

You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:

```
RcpCommand sendRcp:={
Command("rcp.xml?command=0x002f&type=P_STRING&direction=WRITE&num=1")
SSL(true) Port (HTTPS) IP ("192.168.0.1") Password ("anonymous") Username (User)
Name ("Rcp Command 0x002")};
```

5.1.7**VCA Configuration**

With this action you can apply a certain VCA configuration profile to a video line.

Standard parameter:

- Line(<line>)

Enter the video line (starting from 1) for which you would like to set the configuration.
- Configuration(<config>)

Enter the number of the configuration profile which you would like to set. '0' means no configuration.

Optional parameter:

- Name("Action Name")

You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Example:

```
VCAConfiguration setVCAConfig:={
Line(1) Configuration(1) Name("SetConfig 1")};
```

5.1.8**Operation Mode**

You can configure the operation mode of C-states. C-states have a normal behavior. But you can change the mode with this action. You can configure the operation mode of C-states only. You can specify bistable, monostable and periodic mode.

**NOTICE!**

If you change the script, all used C-states will be set to the idle state `open`. Further the C-states which were used in the monostable or periodic mode will be set to `disabled`.

Bistable

In the bistable mode you can specify whether the C-state should be open or closed.

Standard parameter:

- Idle(<idle state>)

Configure the <idle state> here. It can be `open` or `closed`. The default value for all states is `open`.

Monostable

In the monostable mode, the C-state will return to `disable` after a defined time.

Standard parameter:

- High(<time>)

Enter the duration of <time> in tenths of a second. `High` means, the C-state is set to enabled and after the time it will be set to disabled. The time must not be 0.

Optional parameter:

- Idle (<idle state>)
Configure the <idle state> here. It can be open or closed. The default value for all states is open.

Periodic

In the periodic mode can be defined how long a C-state should be enabled or disabled.

Standard parameters:

- Low(<time>)
Enter the duration of <time> in one tenth of a second. Low means, the C-state is set to disabled. The <time> value must not be 0.
- High(<time>)
Enter the duration of <time> in one tenth of a second. High means, the C-state is set to enabled. The <time> value must not be 0.

Optional parameter:

- Idle (<idle state>)
Configure the <idle state> here. It can be open or closed. The default value for all states is open.
- Name("Action Name")
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

Examples:

```
OperationMode bistable:={Idle(closed)Name("bistable mode")};
OperationMode monostable:={High(20)};
OperationMode periodic:={Low(100)High(20)Idle(closed)};
```

The allocation of a configured operation mode will be explained in detail in *Section 5.3.3 State Operation Mode*, page 17.

5.1.9**Timer**

This action is used for setting the daily or weekly timer. So you can define actions that are executed at a particular time. A day is subdivided into 24 time slices. The defined time is evaluated every minute. Therefore the time can be defined from 00:00 to 23:59. The current state of the action is interpreted by the identifier of the action. The kind of polling the current state of the timer is explained in *Section 5.2.3 Action States*, page 15.

Standard parameters:

- TimeBegin(<time>)
Enter the start <time> of the timer action, for instance 11:01.
- TimeEnd(<time>)
Enter the end <time> of the timer action, for instance 23:01. If the end time is set before the start time, the timer is disabled from the end time to the start time. Example:
TimeBegin(22:10)TimeEnd(10:10). The timer is enabled from 22:10 to 23:59 and from 00:00 to 10:10 at the same day. The other time the timer is disabled. If you use the optional parameter, this condition will depend on the weekdays.

Optional parameter:

- DayBegin (<weekday>)
Here you enter the <weekday> in a short form (Mo, Tu, We, Th, Fr, Sa and Su) when the timer should start.
- DayEnd (<weekday>)
Here you enter the <weekday> in a short form (Mo, Tu, We, Th, Fr, Sa and Su) when the timer should end.
- Name ("Action Name")
You can enter a name for the action here. This name is then recorded as meta data when the action is processed. Later, this name can, for example, be searched for in recordings.

**NOTICE!**

If you want to define a weekly timer you must set both parameters DayBegin and DayEnd.

Examples:

```
Timer daily:={TimeBegin(01:13)TimeEnd(14:30)};
Timer weekly:={
TimeBegin(01:13)TimeEnd(14:30)DayBegin(Mo)DayEnd(Th)Name("weekly timer");
```

5.1.10**Stopping of an Action**

Some actions have the particular property when they are activated, then they will be executed permanently, for instance recording. The script language supports the possibility to define a statement for stopping the recording or disconnecting an activated connection and so on. The common syntax is:

```
Stop(<identifier>)
```

The <identifier> has to belong to a defined action. In many cases use of the stop property is not obvious. In the following list all actions are shown where a stop property can be used and it also explains the meaning of them:

- Recording
Stop the recording.
- ConnectionList
If the parameter AutoConnect was set to true but no connection could be successfully established. This action will keep running until a connection has been established. The action can be terminated with the stop command at any time.
- Connect
Disconnect the connection.

The action operation mode (see *Section 5.1.8 Operation Mode*, page 11) has a different syntax. Here, you must allocate the stopping mode the C-state as follows:

```
<C-state>:=Stop(<identifier>)
```

The <identifier> has to belong to a defined action operation mode. If a C-state is deactivated, the idle state is always set to open. Furthermore the C-states which were used in the monostable and periodic mode will be set to disabled. So after a deactivation the C-states have always a defined state.

5.2 State Types

There are different state types. With the help of the script language it is possible to request the value of a state which is a Boolean. The general syntax of the different state types is:

- <state name>
- <state name>(<integer>)
- <state name>(<identifier>)
- <state name>(<integer>,<integer>)

5.2.1

I/O-States

The I/O-states are R-states or C-states, which are as follows:

R-states:

- HddError
Triggered by an HDD (Hard Disk Drive) defect.
- Connect
Triggered whenever a connection has been established.
- EthernetLink(<index>)
There are internal and external links. The internal link has always the index 0. If a device only has one link, so the internal link is coevally the external link. And if a device has additionally a fiber link, it always has the highest index of the links. An activated link status is always 1 or true.
- Input(<index>)
Triggered by external alarm input.
- VideoAlarm(<camera>)
Triggered by interruption of the video signal.
- Motion(<camera>)
Triggered by motion alarm.
- Remote(<index>)
Triggered by remote station's switching contact (only if a connection exists).
- VCARule(<camera>,<rule>,<configuration(opt)>)
Triggered by a certain VCA alarm (rule) from a video line (camera). The <configuration> parameter is optional and checks additionally if the alarm was generated by the given VCA configuration profile. See also the documentation *IVA Task Script Language*.
- Key(<index>)
Triggered by a Softkey (1-8) or the Alarm Key (0) of a connected IntuiKey keyboard.
- NightMode
Triggered when the night mode of the device is activated. (Only for devices supporting night mode.)
- AudioAlarm(<audio line>)
Triggered by an audio alarm on the corresponding audio line.
- VitualAlarm(<index>)
Triggered by a virtual alarm input.
- IsFirstPass
Triggered once when the Alarm Task Engine is started and the script is processed for the first time. This can be used e.g. to initialize other states.

- `VCA_TaskRunning (<camera>)`
Triggered when the VCA task of a video line (camera) is running. This state can, for example, be used to perform an action or an alarm in case the VCA task is interrupted.

C-states:

- `Relay (<index>)`
The number of the relays depends on the device.
- `TempState (<index>)`
There are 18 temporal states for saving.

**NOTICE!**

The number of cameras and indices depends on the particular device except the temporal states. All numbers start with 1 except the index of the Ethernet links which starts with 0.

5.2.2**Tamper States**

The MOTION+, IVMD (Intelligent Video Motion Detection) or IVA algorithms (Intelligent Video Analysis) can detect tampering of a camera. The default algorithm is MOTION+. The different kinds of tampering need not be enabled from the configuration of the algorithms except the tamper state `RefImageCheckFailed`. Configuring of the tamper states is also possible on the browser page: Settings > Alarm > VCA. You can disable or enable all tamper states here. The output of possible tamper actions is accessible via Boolean states. For each state the parameter is `<camera>`. So you have to enter the according camera for the tamper state. The following tamper states are available:

- `SignalTooNoisy (<camera>)`
If the video signal becomes too noisy, in such a way that the automatic object detection becomes impossible then the state will be enabled. This can happen if an analog video signal is transmitted over a large distance.
- `SignalTooBright (<camera>)`
If the video signal becomes too bright, so that the automatic object detection becomes impossible then the state will be enabled. This can happen if the camera is dazzled by a strong light source.
- `SignalTooDark (<camera>)`
If the video signal becomes too dark, so that the automatic object detection becomes impossible then the state will be enabled. This can happen if the camera is covered by a sheet, in such a way that an almost black image is recorded.
- `SignalLoss (<camera>)`
If the video signal is lost, the state is enabled.
- `RefImageCheckFailed (<camera>)`
If a reference image has been set during the configuration of the algorithms and the reference checking of the algorithms has been enabled, the manipulation of the camera is detected by comparing the current video signal with the preset reference image. Significant differences between the two images are forwarded to the Alarm Task Engine.
- `GlobalChange (<camera>)`
If most of the image content has been changed, the state is enabled. This can happen if the camera is moved or if an object comes too close to the camera.

5.2.3**Action States**

Some actions have R- or R/W-states. Therewith, you can verify whether a defined action was successful or not. For instance, if you have defined an e-mail but the SMTP server does not

work or the login is wrong, the e-mail will not be sent. Such errors can be evaluated with the following syntax:

```
Error(<identifier>)
```

This error state is an R/W-state and the <identifier> has to belong to a configured action, for instance `Error(mail)`. Here, mail is the identifier of the action `AlarmMail` (see *Section 5.1.1 Send Alarm E-Mail*, page 4). But the type of the errors cannot be interpreted. That means, you cannot specify the source of failure. The following list shows the actions which have an error state and some hints why an action could not be successfully executed:

- `RcpCommand`
The sending of an RCP command was not successful. Maybe you have entered a wrong IP address.
- `Connection`
The connection was not successfully established, e.g. because the IP address was incorrect.
- `AlarmMail`
The mail could not be sent to an SMTP server, e.g. because the server was password protected.
- `JpegPosting`
The posting to an FTP server was not successful because the server could not be reached.
- `VCAConfiguration`
Setting the VCA configuration was not successful.



NOTICE!

Error states are not set back to disabled.

That means, if the action, for example JPEG posting, was not successful, the error state `Error(posting)`, will not be set back to disabled. So, if you execute the action once more and the posting was not successful again, you cannot evaluate the error. It is your responsibility to set back the error state, for instance:

```
JpegPosting posting:={IP("192.168.0.1")
Login("anonymous")Password("anonymous")};
if(Error(posting))then Error(posting):=false;
```

Furthermore you can verify whether a defined action is activated or not, for instance, with the action `Connection`. The syntax of this state is:

```
IsActivated(<identifier>)
```

The <identifier> has to belong to a configured action as follows:

- `Connection`
If there is a connection to the entered IP address, the state is enabled else disabled.
- `ConnectionList`
By trying to connect an encoder or decoder the state is enabled else disabled.
- `Timer`
If the configured timer is active the state is also enabled else the state is disabled.

A practical example for using a timer could be: A relay should be enabled from 8:00 a.m. to 6:00 p.m. else disabled. First, you must define and configure a timer this way:

```
Timer dailyRelay:={TimeBegin(08:00)TimeEnd(18:00)};
if(IsActivated(dailyRelay))then Relay(1):=true else Relay(1):= false;
```

5.3 Conditions

5.3.1 Boolean Composition of Conditions

A Boolean condition is a condition that returns a Boolean value, that is `true` or `false`. In this Boolean composition you can use:

- constants
- negation
- conjunction
- disjunction

There are two constants which have a value with a fixed meaning. These constants are `true` (1) and `false` (0). The syntax of negation, conjunction and disjunction is:

The negation of conditions is:

```
<condition>:=!<condition>
<condition>:=not <condition>
```

The conjunction of conditions is:

```
<condition>:=<condition> && <condition>
<condition>:=<condition> and <condition>
```

The disjunction of conditions is:

```
<condition>:=<condition> || <condition>
<condition>:=<condition> or <condition>
```

Ambiguities in the evaluation of expressions are resolved by priorities. Negations have the highest priority, followed by conjunctions, and last disjunctions. Furthermore, the priority can be controlled by embracing sub-expressions with brackets:

```
<condition>:=(<condition>)
```

With the help of the Boolean conditions you can combine all R-, R/W- and C-states with each other.

Examples:

- `(Relay(1) || Motion(2)) && VideoAlarm(1)`
- `IsActivated(con_1) and VCARule(2,3)`
- `!Connect or Error(sendRcp)`

5.3.2 State Condition

With a state condition it is possible to set an R/W-state depending on a Boolean condition.

The general syntax is:

```
<R/W-state>:=<condition>;
```

Consequently, the `<R/W-state>` is defined by the `<condition>` on the right. If the result of the condition is true, the R/W-state is enabled and if the condition is false the R/W-state is disabled. Note that the R/W-state is only set if this one has changed. Some practical examples for using states:

Examples:

```
Relay(1):=Motion(2) && VideoAlarm(1);
TempState(2):=not Relay(2) and VCARule(2,3);
Relay(2):=true;
```

5.3.3 State Operation Mode

The C-states `TempState(<index>)` and `Relay(<index>)` have a special property. It is possible to allocate a specific operation mode like bistable, monostable or periodic. Before an operation mode is allocated, this one must be defined. Such statements will execute only once if the Alarm Task Engine is initialized. The syntax is:

```
<C-state>:=<identifier>;
```

The <identifier> must be of the action `OperationMode`. The following example shows the kind of configuring of a C-state. In this example a relay is disabled for 10 seconds and enabled for 2 seconds:

```
OperationMode periodic:={Low(100)High(20)};
Relay(1):=periodic;
```

In this way you can just activate the operation mode of a C-state. It is not possible to stop or rather to deactivate the C-state in such a way.

Of course, you can stop the operation mode of a C-state, but this will be explicated in the next chapter.

5.3.4

Action Condition

With an action condition defined actions can be executed on a particular condition. The syntax of an action condition is:

```
if(<condition>)then <statement list> [else <statement list>];
```

A <statement list> is defined as:

```
<statement_list>:=<statement>
                |<statement>,<statement list>

<statement>:=<identifier>
            |<state condition>
            |<state operation mode>
```

The <identifier> has to belong to a configured action, the <state condition> is explained in *Section 5.3.2 State Condition*, page 17 and <state operation mode> in *Section 5.3.3 State Operation Mode*, page 17.

For instance, you have configured an action for recording and the recording should start if the relay is enabled. So, you must define an action condition as follows:

```
Recording recording:={Camera(1)};
if(Relay(1))then recording;
```

Another practical example with more configured actions could be the sending of an alarm e-mail and posting of a JPEG-file after a motion alarm. First, you have to configure both actions and then you have to define the action condition this way:

```
JpegPosting posting:={IP("192.168.0.1")
Login("anonymous")Password("anonymous")};
AlarmMail mail:={IP("192.168.0.1")To("to@vcs.com")};
if(Motion(1))then mail,posting;
```

Yet another example could be a relay set to monostable mode from 8:00 a.m. to 10:00 p.m.

The other time of day the relay should be set to a normal mode. Now, you have to configure an operation mode, a timer and a condition action:

```
OperationMode monostable:={High(50)};
Timer daily:={TimeBegin(8:00)TimeEnd(22:00)};
if(IsActivated(daily))then Relay(1):=monostable
else Relay(1):=Stop(monostable);
```



NOTICE!

The statements will be evaluated if the Boolean condition alternates from disabled to enabled or from enabled to disabled.

If for security reasons you want to send an alarm e-mail each time one of the relays `Relay(1)` or `Relay(2)` is enabled, you must define the action condition this way:

```
if(Relay(1))then sendMail;if(Relay(2))then sendMail;
```

You cannot define the action condition in the following way:

```
if(Relay(1) or Relay(2))then sendMail;
```

In this action condition the alarm mail is only sent once even if both relays are enabled.

5.4 Comments

You can set comments within the script so that you can describe your actions or states or comment these out. There are two comment types. The first one is the line comment. It starts with the characters `//` (double slash) followed by other characters and it ends with a new line like:

```
//comment
```

The other one is the multiline comment. This one begins with `/*` (slash and asterisk) and it ends with `*/` (asterisk and slash) as follows:

```
/*multiline  
comment*/
```

It is also possible to mix both types. Everything within a comment is not evaluated. But since the script size is restricted you should limit the number of comments (see *Section 4 Configuration*, page 2).

6 Starting the Alarm Task Engine

If you have created a script automatically or manually, the script will be sent to the device. It is recommended to send the script via HTTPS. Otherwise the passwords and login information are sent plain. Then the script is parsed on the device. If the script does not have a syntax error, the message: "Script successfully parsed" is displayed above the Alarm Task Editor and a new Alarm Task Engine is created. If there is an older Alarm Task Engine that one will run until the new one is created. Before the new Alarm Task Engine starts to run, all previously created events, which belong to the older Alarm Task Engine, are deleted. All actions are also deleted except the one which is just being executed. Furthermore, the temporal states (see *Section 5.2.1 I/O-States*, page 14) will be set to `disabled` and then the Alarm Task Engine is initialized, for instance, the operation mode of a C-state is activated or actions are being executed when the Boolean condition is `true` and so on. For instance, you have configured the device and it is going to reset. After the restart the device can reconnect automatically or send an alarm e-mail immediately this way:

```
Connection con_1:={IP("192.168.9.1")};  
AlarmMail restart:={IP("www.mailserver.de")  
To("test@vcs.com") Subject("restart")};  
if(true)then restart,con_1;
```

7 Glossary

C

C-state	Configurable state
CGI	Common Gateway Interface
CIF	Common Intermediate Format

F

FTP	File Transfer Protocol
-----	------------------------

H

HDD	Hard Disk Drive
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure

I

I/O	Input/Output
IP	Internet Protocol

J

JPEG	Joint Photographic Expert Group
------	---------------------------------

M

MPEG	Moving Picture Experts Group
------	------------------------------

Q

QCIF	Quarter Common Intermediate Format
------	------------------------------------

R

R-state	Readable state
RCP	Remote Control Protocol
R/W-state	Readable/Writable-state

S

SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SSL	Secure Socket Layer

T

TCP Transmission Control Protocol

U

URL Uniform Resource Locator

Bosch Sicherheitssysteme GmbH

Werner-von-Siemens-Ring 10

85630 Grasbrunn

Germany

www.boschsecurity.com

© Bosch Sicherheitssysteme GmbH , 2010