

# Alarm Task Script Language

Version 4.2



**BOSCH**

en Script Language



# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Definitions</b>	<b>5</b>
2.1	Actions	5
2.2	Events and States	5
2.3	Alarm Task Engine	5
<b>3</b>	<b>System Integration</b>	<b>6</b>
<b>4</b>	<b>Configuration</b>	<b>7</b>
<b>5</b>	<b>Syntax</b>	<b>9</b>
5.1	Action Types	9
5.1.1	Send Alarm E-Mail	9
5.1.2	JPEG Posting	10
5.1.3	Recording	12
5.1.4	Connection	12
5.1.5	Connection List	14
5.1.6	RCP Command	15
5.1.7	VCA Configuration	16
5.1.8	Bicom Command	16
5.1.9	Control Code	17
5.1.10	Message Check	17
5.1.11	Operation Mode	19
5.1.12	Timer	20
5.1.13	Stopping an Action	20
5.2	State Types	21
5.2.1	I/O States	21
5.2.2	Tamper States	22
5.2.3	Action States	23
5.3	Conditions	25
5.3.1	Boolean Composition of Conditions	25
5.3.2	State Condition	25
5.3.3	State Operation Mode	25
5.3.4	Action Condition	26
5.4	Comments	27
<b>6</b>	<b>Starting the Alarm Task Engine</b>	<b>28</b>
	<b>Glossary</b>	<b>29</b>

# 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, sending of RCP commands 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 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 number of I/O states depends 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 action states are distinguished between:

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

R-state means, you can just query its current value. If you use R/W-states, you can change the value. C-states can be used to set it to a 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.



## 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 on the \Settings.html browser page:

- 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 on the \Settings.html page:

- Alarm > Alarm Task Editor

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

**Figure 4.1** Alarm E-Mail Dialog Box

Figure 4.2 displays the automatically created script. You can see in the script that an alarm e-mail will be sent to the entered mail server IP 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 manually. 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.

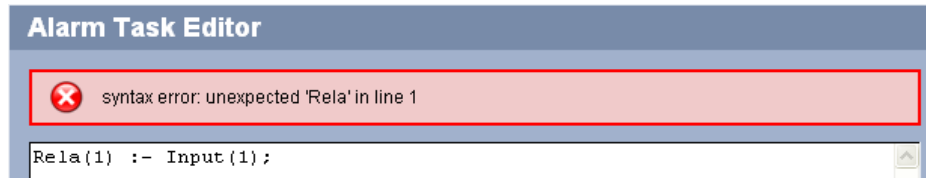
If you edit the script manually and the script contains syntax errors, each error will be listed in the message box above the script editor, indicating the corresponding line. See Figure 4.3.

Possible syntax errors are:

- password or URL too long
- missing bracket, semicolon, and so on

The message box could also display other warnings.

If the script is free of syntax errors, the message: "Script successfully parsed." appears.



**Figure 4.3** Alarm Task Engine Error Message



**NOTICE!**

The maximum size of the zipped script file is about 4 KB. 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>]};
```

There are two different types of actions, asynchronous actions and synchronous actions. It can take several seconds before asynchronous actions are finished or started. 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
- BicomCommand
- ControlCode
- MessageCheck

#### Synchronous:

- OperationMode
- Timer

An <identifier> begins with a lower-case phrase that can be followed by either:

- A capitalized phrase
- A lower-case phrase
- A digit
- An underscore

#### Example:

```
alarmMail_1
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. Thus 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 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`: 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 `videoserver`.
- `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,...)`  
In order to select 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")`  
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 individually posting one or more JPEG images on an FTP server.

**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. You can choose between three <formats>:
  - Small: 176×144/120 pixels (QCIF)
  - Medium: 352×288/240 pixels (CIF). This is the default value.
  - 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 files 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 255 is reached, the counting process will start again from 000. If you modify the script, the suffix will also start from 000. If the server is not available, the number will be increased nonetheless. This means that the generated file names with these numbers will 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 set correctly. 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. This action has one standard parameter only. Preconditions for recording are a storage medium and that the recording scheduler is enabled.

Further reading:

*Section 5.1.13 Stopping an Action, page 20*

**NOTICE!**

If you change the script, the activated recording will not be stopped because the recording might have been 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 vice versa. 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.

Further reading:

*Section 5.1.13 Stopping an Action, page 20*

**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, the default value. It is recommended to use the UDP protocol.
- TCP

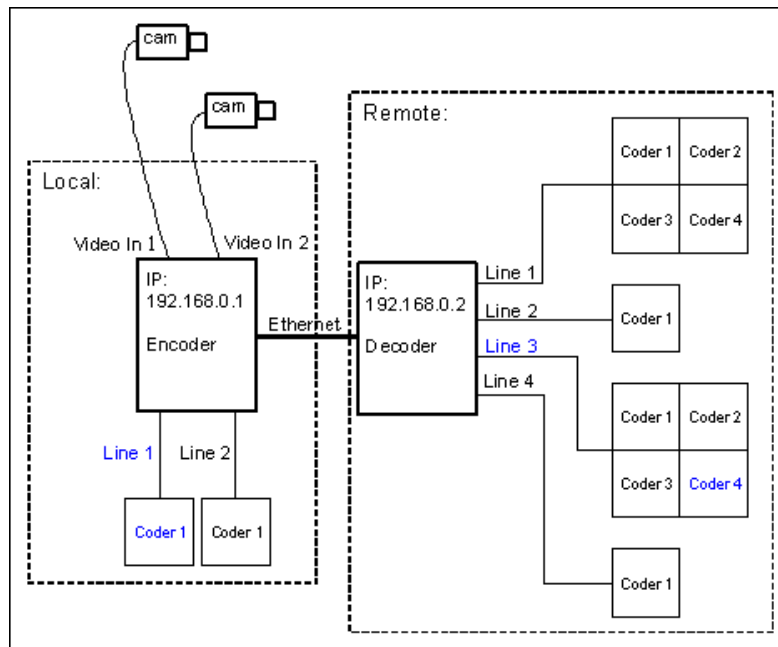
- LocalLine (<line>)

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`: 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 SSL 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 a HTTPS connection, you must enable SSL mode. The default value for SSL mode is `false`.
- `Audio(<boolean>)`  
For `<boolean>`, enter `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`
  - `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 replace 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 called 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 in which connections should be activated until a connection has been established. It is also possible to stop this action.

Further reading:

*Section 5.1.13 Stopping an Action, page 20*

**Standard parameter:**

- `Connection(con_1,con_2,...)`  
List the connections here. Before you enter the connections, you must define all of them. The action will try to connect the first one after about 10 seconds and proceed with the next one until a connection has been established.

Further reading:

*Section 5.1.4 Connection, page 12*

**Optional parameters:**

- `AutoConnect(<boolean>)`  
You can enable the `AutoConnect` property with `true` or disable it with `false` (default). If the property is activated, the connection list is restarted after one second delay until a connection has been established. 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. 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 parameters:**

- `SSL(<boolean>)`  
With `true` you can 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 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 authorization level 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 parameters:**

- 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****Bicom Command**

With this action you can send Bicom commands to the device itself (local host). For further information about Bicom, please see the BiCom–Bilinx Command Interface documentation.

**Standard parameters:**

- Server (<BicomServer>)
 

Enter the <BicomServer> to which the command should be sent here:

  - DeviceSrv: Device Server
  - CamersSrv: Camera Server
  - PtzSrv: PTZ Server
  - CaSrv: Content Analysis Server
  - IoSrv: I/O Server
- ObjectId(<Id>)
 

Enter the Bicom object <Id> here. The ID can be either decimal or hexadecimal.
- Payload("0x1234")
 

Enter the Bicom command payload here. The payload must be hexadecimal and start with '0x'. The length of the payload must match with the length specified in the Bicom documentation for the corresponding object.

**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:**

```
BicomCommand sendBicom:={
Server(CameraSrv)ObjectId(0x0140)Payload("0x0001")Name("Bicom Command
0x0140")};
```



## 5.1.9

### Control Code

With this action you can send (Bicom-) control codes to the device itself (local host). For further information about Bicom control codes please see the BiCom–Bilinx Command Interface documentation.

#### Standard parameters:

- Operation(<type>)
  - Enter the <type> of operation here:
    - AuxOn
    - AuxOff
    - Shot
    - Set
- Number(<num>)
  - Enter the control code number <num> here.

#### 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:

```
ControlCode commandLock:={
  Operation(AuxOn)Number(90)Name("ControlCode 90 on");
}
```

## 5.1.10

### Message Check

With this action you can check for received messages from the device itself (local host) or from another device. The message check needs to be performed regularly and within a time period shorter than 20 seconds. Otherwise messages can be lost. Additionally, a maximum of 32 messages can be received between two consecutive checks. If the specified message has been received, the action state is set to 'activated', which can be checked via 'IsActivated()'.

#### Standard parameter:

- Message("<message>")
  - Here you can specify which messages you like to check.
  - <message> has the same structure like the CGI command.
  - It is composed as follows:
    - 'rcp.xml?message=' plus a message tag code or a list of message tag codes

#### Examples:

```
rcp.xml?message=0x01C0
rcp.xml?message=0x01C0$01C1
```

#### Optional parameters:

- Payload("0x1234")
  - Here you can specify a certain payload. This is checked against the payload which has been received within the message. Only if the message has the same payload as specified here, the MessageCheck action state is activated.
- Num(<index>)
  - Here you can specify a certain numeric parameter. This is checked against the numeric parameter which has been received within the message. Only if the message has the same numeric parameter as specified, the MessageCheck action state is activated.

- SSL (<boolean>  
With `true`, you can enable or with `false` disable the encryption mode for sending the message check.
- Port (<port>  
Enter the <port> for sending the message check 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 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 authorization level 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:**

```
MessageCheck msg1 := {
Message ("rcp.xml?message=0x0a8b") Num(1) Payload("0x00") IP("192.168.0.1")};
```

```
OperationMode trigger5sec := { Low(1) High(50) };
```

```
TempState(1) := trigger5sec;
```

```
if(TempState(1)) then msg1;
```

```
if( IsActivated(msg1) ) then Relay(1):=true;
```

**Further reading:**

*Section 5.2.3 Action States, page 23*

### 5.1.11

#### 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 set a bistable, monostable or periodic mode.



#### NOTICE!

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

#### Bistable

With 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 monostable mode, the C-state will return to disabled after a defined time.

##### Standard parameter:

- `High(<time>)`  
Enter the duration of `<time>` in tenths of a second. `High` means, the C-state is enabled and after the specified time it will be 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

With periodic mode, you can define how long a C-state should be enabled or disabled.

##### Standard parameters:

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

##### Optional parameters:

- `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)};
```

Further reading:

*Section 5.3.3 State Operation Mode, page 25.*

**5.1.12****Timer**

This action is used for setting the daily or weekly timer. Thus 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. Thus 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.

Further reading:

*Section 5.2.3 Action States, page 23.*

**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 00:00 to 10:10 and from 22:10 to 23:59 at the same day. From 10:11 to 22:09, the timer is disabled. If you use the optional parameters, this condition will depend on the weekday.

**Optional parameters:**

- DayBegin(<weekday>)  
Here you enter the <weekday> in a short form (Mo, Tu, We, Th, Fr, Sa and Su) on which the timer should start.
- DayEnd(<weekday>)  
Here you enter the <weekday> in a short form (Mo, Tu, We, Th, Fr, Sa and Su) on which 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")};
```

**5.1.13****Stopping an Action**

Some actions have the particular property of being executed permanently once they have been activated, for instance the Recording action. The script language enables you to define a statement for stopping such an action, for example stopping a recording or disconnecting an activated connection. The common syntax is:

Stop(<identifier>)

The <identifier> has to belong to a defined action. In many cases, using of the Stop command is not obvious. The following list shows all actions where the Stop command can be used and explains the meaning:

- Recording  
Stops the recording.
- ConnectionList  
If the AutoConnect parameter was set to true but no connection could be 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  
Disconnects the connection.

The operation mode action has a different syntax. Here, you must allocate the Stop command 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. Thus, after being deactivated, C-states have always a defined state.

Further reading:

*Section 5.1.11 Operation Mode, page 19*

## 5.2 State Types

There are different state types. The script language enables you 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

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

#### R-states:

- HddError  
Triggered by an HDD 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, the internal link is coevally the external link. If a device additionally has 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 an external alarm input.
- VideoAlarm(<camera>)  
Triggered by an 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 specified VCA configuration profile. See also the IVA Task Script Language documentation.
- 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.
- VirtualAlarm(<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, for example, to initialize other states.
- VCATaskRunning(<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.
- ManipulationAlarm(<index>  
Triggered by an external manipulation alarm (for example if the housing of the device is opened). Only for devices which support manipulation alarms.

**C-states:**

- Relay(<index>  
The number of 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 (Intelligent Video Analysis) algorithms can detect tampering of a camera. The default algorithm is MOTION+. The different kinds of tampering do not need to be enabled from the configuration of the algorithms except the RefImageCheckFailed tamper state . Configuring the tamper states is also possible on the browser page under Settings > Advanced Mode > 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>. Thus you have to enter the corresponding camera for the tamper state. The following tamper states are available:

- SignalTooNoisy(<camera>  
This state will be enabled if the video signal becomes too noisy, in such a way that automatic object detection becomes impossible. This can happen if an analog video signal is transmitted over a large distance.

- `SignalTooBright (<camera>)`  
This state will be enabled if the video signal becomes too bright, so that automatic object detection becomes impossible. This can happen if the camera is dazzled by a strong light source.
- `SignalTooDark (<camera>)`  
This state will be enabled if the video signal becomes too dark, so that automatic object detection becomes impossible. This can happen if the camera is covered by a sheet, in such a way that an almost black image is recorded.
- `SignalLoss (<camera>)`  
This state will be enabled if the video signal is lost.
- `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, 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>)`  
This state will be enabled if most of the image content has been changed. 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`. 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`  
Sending an RCP command has not been successful. Maybe you have entered a wrong IP address.
- `Connection`  
The connection has not been established, e.g. because the IP address is incorrect.
- `AlarmMail`  
The mail could not be sent to an SMTP server, e.g. because the server is password protected.
- `JpegPosting`  
Posting to an FTP server has not been successful because the server could not be reached.
- `VCAConfiguration`  
Setting the VCA configuration has not been successful.
- `BicomCommand`  
Sending a Bicom command has not been successful. Maybe the object ID is wrong or maybe the payload length is incorrect.
- `ControlCode`  
Sending a control code has not been successful. Maybe the control code number is incorrect or out of range.

- MessageCheck  
The message check has not been successful. Maybe the time interval between two checks was too long or the remote device is not available.

**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. Thus, if you execute the action once more and the posting was not successful again, you cannot evaluate the error. Therefore, you must set back the error state.

**Example:**

```
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 `Connection` action. 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 IP address entered, the state is enabled, otherwise it is disabled.
- ConnectionList  
When trying to connect an encoder or decoder, the state is enabled, otherwise it is disabled.
- Timer  
If the configured timer is active, the state is enabled, otherwise it is disabled.

**Example:**

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

- MessageCheck  
If the defined message has been received and - if specified - the payload and numeric parameters are correct, the state is enabled. After a short time the state is automatically disabled again.



## 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:

- Negation of conditions:
  - `<condition>:=!<condition>`
  - `<condition>:=not <condition>`
- Conjunction of conditions:
  - `<condition>:=<condition> && <condition>`
  - `<condition>:=<condition> and <condition>`
- Disjunction of conditions:
  - `<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 disjunctions have the lowest priority.

Furthermore, the priority can be controlled by putting sub-expressions in parentheses:

```
<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 `TempState(<index>)` and `Relay(<index>)` C-states have a special property. It is possible to allocate them a specific operation mode like bistable, monostable or periodic.

Before an operation mode is allocated, it 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 an OperationMode action. The following example shows how to configure 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;
```

This way, you only can activate the operation mode of a C-state. It is not possible to stop or to deactivate the C-state this way.

Further reading:

*Section 5.3.4 Action Condition, page 26*

## 5.3.4

### Action Condition

Actions defined with an action condition 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> must belong to a configured action.

#### Examples:

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

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

After a motion alarm, an alarm e-mail should be sent and a JPEG-file should be posted. First, you have to configure both actions, then you have to define the action condition as follows:

```
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;
```

A relay should be set to monostable mode from 8:00 a.m. to 10:00 p.m., the remaining time the relay should be set to normal mode. Therefore, 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 as follows:

```
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.

Further reading:

*Section 5.3.2 State Condition, page 25*

*Section 5.3.3 State Operation Mode, page 25*

## 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 types of comments.

The first one is the line comment. It starts with the characters `//` (double slash) followed by other characters and ends with a new line like:

```
//comment
```

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

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

It is also possible to combine both types. Everything within a comment is not evaluated. Since the script size is restricted, you should limit the number of comments to the necessary extent.

Further reading:

*Section 4 Configuration, page 7*

## 6 Starting the Alarm Task Engine

If you have automatically or manually created a script, 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 is free of syntax errors, 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, it will run until a new one is created. Before the new Alarm Task Engine starts to run, all previously created events which belong to an older Alarm Task Engine are deleted. Also, all actions are deleted except the one which is being executed. Furthermore, the temporal states 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 immediately send an alarm e-mail :

```
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;
```

Further reading:

*Section 5.2.1 I/O States, page 21*

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

---

### S

---

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

---

---

**T**

---

TCP	Transmission Control Protocol
-----	-------------------------------

**U**

---

UDP	User Datagram Protocol
-----	------------------------

---

URL	Uniform Resource Locator
-----	--------------------------



**Bosch Security Systems**

Werner-von-Siemens-Ring 10  
85630 Grasbrunn  
Germany

**[www.boschsecurity.com](http://www.boschsecurity.com)**

© Bosch Security Systems, 2010