2N® Helios IP Automation

Configuration Manual

Version 2.5.0
Firmware 2.5

www.2n.cz

The 2N TELEKOMUNIKACE a.s. joint-stock company is a Czech manufacturer and supplier of telecommunications equipment.

The product family developed by 2N TELEKOMUNIKACE a.s. includes GSM gateways, private branch exchanges (PBX), and door and lift communicators.
2N TELEKOMUNIKACE a.s. has been ranked among the Czech top companies for years and represented a symbol of stability and prosperity on the telecommunications market for almost two decades. At present, we export our products into over 120 countries worldwide and have exclusive distributors on all continents.

2N TELEKOMUNIKACE administers the FAQ database to help you quickly find information and to answer your questions about 2N products and services. On faq.2n.cz you can find information regarding products adjustment and instructions for optimum use and procedures „What to do if...".

Declaration of Conformity

2N TELEKOMUNIKACE a.s. hereby declares that the 2N® Helios product complies with all basic requirements and other relevant provisions of the 1999/5/EC directive. For the full wording of the Declaration of Conformity see the CD-ROM enclosed and at www.2n.cz.

The 2N TELEKOMUNIKACE company is a holder of the ISO 9001:2008 certificate. All development, production and distribution processes of the company are managed by this standard and guarantee a high quality and advanced technical level of and a professional approach to all of our products.

Table of Contents

1 2N® Helios IP Automation

In this section, we describe the **2N® Helios IP Automation** configuration.

Here is what you can find in this section:

- Terms and Symbols
- 2N® Helios IP Automation Configuration
- Description of Configurable Blocks

1.1 Terms and Symbols

Symbols Used in Manual

Safety

- **Always** abide by this information to prevent injury of persons.

Warning

- **Always** abide by this information to prevent damage to the device.

Caution

- **Important information** for system functionality.

Tip

- Useful advice for quick and efficient functionality.

Note

- Routines or advice for efficient use of the device.

1.2 2N® Helios IP Automation Configuration

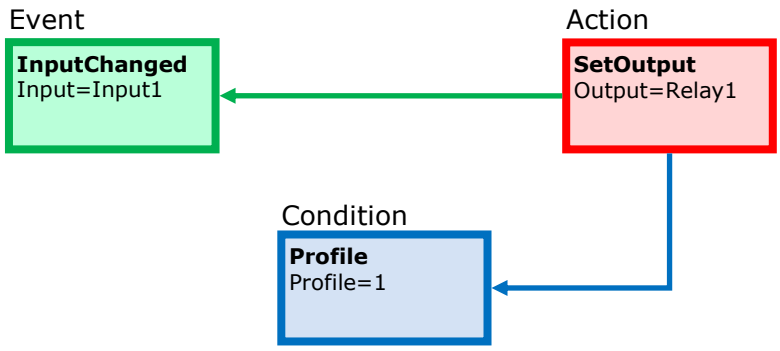
2N® Helios IP provides flexible setting options depending on the user's requirements. If the standard setting options (switch/call settings, e.g.) are insufficient for the intended use, apply a special programmable interface - 2N® Helios IP Automation. Typically, 2N® Helios IP Automation is helpful for applications that require rather complex interconnection with the third parties' systems.

Note

- 2N® Helios IP Automation works only with a valid **Enhanced Integration** or **Gold** licence key.

Some 2N® Helios IP models are equipped with a number of digital inputs and outputs, most of which can be configured like standard 2N® Helios IP switches (refer to the Switches subsection). You can make use of all of these 2N® Helios IP Automation inputs and outputs in variable combinations.

2N® Helios IP Automation helps you combine the **Event**s arising in the system (such as key pressing, RFID card use, digital input status change, etc.) with specific **Action**s (such as digital output activation, user sound playing, call, etc.) as necessary. Moreover, the execution of actions can be bound by selected **Condition**s (time profile state, logic input state, e.g.).



The figure above shows a typical interconnection of the Event, Action and Condition blocks. It holds true in general that an action is always tied with a selected event and is executed when a selected condition is met. The condition is optional and if none is selected, the action is executed whenever the assigned event occurs. 2N® Helios IP Automation defines a number of events, actions and conditions to be further set. Refer to the subsections below for the full list.

The example shown in the figure above can be interpreted as follows: The **SetOutput** action (digital output setting) is executed if the **InputChanged** event (logic input1 change from log.0 to log. 1) arises and the **Profile** (active profile 1) condition is met.

The 2N® Helios IP web interface helps you configure block combinations (Automation) easily. The configuration shown in the figure below corresponds to the example above.

2N® Helios IP allows up to 12 blocks to be created and interconnected (regardless of the block type - events, actions and conditions). Multiple actions can be assigned to an event or condition. Thus, you can create 6 actions and assign them to 6 events, or create 11 actions and assign them to 1 event, for example.

Block Parameter Settings

Select the required Event (Event.xxx), Action (Action.xxx) or Condition (Condition.xxx) in the **Block type** column. Set one or more parameters for the blocks in the respective row of the **Parameters** column - refer to the block describing subsections below for the supported parameters. Separate the parameters with a colon if more parameters are required.

The changes will not be executed until you press the **Save** button in the right-hand bottom corner of the page.

If you have set a parameter correctly, a green mark will appear at the end of the respective block definition row. If not (if you enter a wrong parameter name/value or fail to complete an obligatory block parameter), a red mark will appear at the end of the row. Move your mouse cursor to the red mark to display the Help to find the error. If all the required blocks have been configured correctly (there is a green mark on every row), 2N® Helios IP Automation will be enabled. If there is a red mark, the 2N® Helios IP Automation function will be disabled.

Most of the blocks include parameters (Event, Condition, StartEvent, e.g.) that refer to other blocks. Set these parameters to interconnect the defined blocks. Make sure that the value to be entered matches the row number in the table defining the block that is referred to. If you enter a wrong value (not matching the defined block type or matching an undefined block) and press **Save**, a red mark will appear at the respective row.

Tip

- The Upper/Lower case need not be respected in the parameter names.
- Some block parameters are optional. If you do not enter an optional parameter in the block definition, the default value will be applied.

Use of Variables

The event block variables (parameters) help transfer additional information between blocks – send ID of the detected card via HTTP to another device, use parameters received in HTTP for setting parameters of a tied action and so on. Their values are updated whenever the event is generated. Use the following syntax to refer to a variable in the configuration parameters of another block:

$(block_number.variable_name) – the block number and variable name are separated with a dot.

Example:

1: Event.KeyPressed: Key=Any

2: Action.SendHttpRequest: Event=1; Uri=http://192.168.1.1/ABCD?Key=$(1.Key)

Press any key (block 1 Event.KeyPressed) to send the HTTP request (block 2 Action.SendHttpRequest) to IP address 192.168.1.1. For example, if you press *, the HTTP request URI will be as follows: **http://192.168.1.1/ABCD?Key=***

Every event defines the **TimeStamp** and **Count** variables.

TimeStamp contains encoded date and time of the last event generation in the Unix Time format (second count from 00:00:00 1.1.1970).

Count contains the count of event generations after the device start or last block configuration change. The variable increases by 1 after each event generation.

Refer to the following subsections for more variables with specific functions.

Tip

- The Upper/Lower case is not be respected in the variable names.

Caution

- You cannot use the variables in the block relation defining parameters, i.e. Event, Condition, etc.

1.3 Events

2N® Helios IP Automation defines the following event types:

KeyPressed	key pressed
CodeEntered	numerical code entered
CardEntered	RFID card entered
DtmfPressed	DTMF code received in call
DtmfEntered	DTMF-received in call numerical code detected
CallStateChanged	call state changed
InputChanged	digital input changed
HttpTrigger	HTTP command received
MulticastTrigger	command for multiple devices received
Delay	delay defined
Timer	periodical event timer

See below for details on the events and their parameters and use.

Event.KeyPressed

The **KeyPressed** block defines the event generated by pressing of the defined key or any key from the defined group.

Parameters

Key

Define the key or a key group. If this parameter is not completed, the event is generated upon pressing of any key (default value: any).

Valid values:
0, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, *, **#** for numerical keypad buttons
%1, **%2**, .., **%54** for quick dial buttons
any for any button (default value).

Separate the values with a comma while defining more keys than one.

SuppressTones

Suppress sound signalling initiated by pressing of a non-programmed quick dial button. The parameter is optional.

Valid values:
0 – tones are not suppressed
1 – tones are suppressed (default value)

Variables

Key

Detected code of the key which was the last to generate this event. The key code is stored in the Key parameter format.

Example

Event generated by pressing of # and quick dial button 3 or 4:

Action.KeyPressed: Key=#, %3, %4

Event.CodeEntered

The **CodeEntered** block defines the event generated by entering of the numerical code and confirmation with the * key (for numerical keypad models only).

Parameters

Code

Define the numerical code.

Valid values:
Numerical code - 12345, e.g.

SuppressTones

Suppress sound signalling initiated by receiving of an invalid numerical code. The parameter is optional.

Valid values:
0 – tones are not suppressed
1 – tones are suppressed (default value)

Variables

Code

Detected numerical code which was the last to generate this event.

Example

Event.CodeEntered: Code=12345

Event.DtmfPressed

The **DtmfPressed** block defines the event that is generated when the defined or any DTMF code is received from the defined group. DTMF codes are detected both in incoming and outgoing calls.

Parameters

Key

Define the DTMF code (or DTMF code group). If this parameter is not completed, the event is generated whenever any DTMF code is detected (default value: Any).

Valid values:
0, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, ***, **#, A, B, C, D**
any for any key (default value).

Separate the values with a comma to specify a group of codes.

Variables

Key

Detected received DTMF code which was the last to generate the event. The DTMF is stored in the Key parameter format.

Example

Event generated upon detection of DTMF code #:

Action.KeyPressed: Key=#

Event.DtmfEntered

The **DtmfEntered** block defines the event that is generated by entering of a DTMF numerical code confirmed with the * key in an incoming or outgoing call.

Parameters

Code

Define the numerical code.

Valid values:
Numerical code – 12345, e.g.

Variables

Code

Detected received numerical code which was the last to generate this event.

Example

Event.DtmfEntered: Code=12345

Event.CardEntered

The **CardEntered** block defines the event generated by entering of the RFID card with the defined ID (for RFID card reader models only).

Parameters

Card

Define the RFID card ID; refer to the Card Reader subsection in the 2N® Helios IP Configuration Manual.

Valid values:

valid – any valid card (included in the intercom card list)
invalid – any invalid card
any – any card

SuppressTones

Suppress sound signalling initiated by detection of an invalid card. The parameter is optional.

Valid values:

0 – tones are not suppressed
1 – tones are suppressed (default value)

Variables

Card

ID of the detected card which was the last to generate this event.

Example

Event generated by entering of a card with ID 0*0012456:

Event.CardEntered: Card=0*0012456

Event.CallStateChanged

The **CallStateChanged** block defines the event generated by a call state change (call ringing, call connection, call termination, etc.)

Parameters

State

Define the call state change.

Valid values:

ringing - ringing start
connected – successful call connection
terminated – call termination

Direction

Define the call direction.

Valid values:

incoming – incoming calls
outgoing – outgoing calls
any – both directions

The parameter is optional, the default value is **any**.

Variables

State

Detected call state which generated this event. The options correspond to the State parameter.

Direction

Detected call direction which generated this event. The options are incoming or outgoing.

Example

Event generated by termination of any outgoing call:

Event.CallStateChanged: State=terminated; Direction=outgoing

Event.InputChanged

The **InputChanged** block defines the event generated by a change of the logic level on the defined digital input.

Parameters

Input

Define the logic input.

Valid values:

tamper – tamper switch input
input1 – digital input 1
input2 – digital input 2
cr_input1 – digital input 1 on card reader
cr_input2 – digital input 2 on card reader

There may be different lists of valid values for different 2N® Helios IP models; refer to the Available Digital Inputs and Outputs subsection.

Edge

Define the detected change on the digital input.

Valid values:

falling – falling edge, change from log. 1 to log. 0
rising – rising edge, change from log. 0 to log. 1

The parameter is optional, the default value is **rising.**

Variables

Input

Detected ID of the input whose change was the last to generate this event. The options correspond to the Input parameter values.

Edge

Detected edge change which was the last to generate this event. The options are falling or rising.

Example

Event generated by disconnection of the tamper switch (device opening):

Event.InputChanged: Input=tamper

Event.Delay

The **Delay** block defines the event generated with a defined delay after another specified event. Define this event to delay the response to the other event by a defined time interval (Delay).

Parameters

StartEvent

Define the event that starts the delay.

StopEvent

Define the event that stops the delay. The parameter is optional.

Delay

Define the delay time.

Example of valid values:

10 – 10 seconds (units are unnecessary)
10s – 10 seconds
100ms – 100 milliseconds

Variables

This block does not define any specific variables.

Example

Event generated 1s after the rise of event on row 1:

Event.Delay: StartEvent=1; Delay=1s

Event.Timer

The **Timer** block defines the event generated with a defined delay after another specified event with a defined count of repetitions. Define this event to delay the response to the other event by a defined time interval, or execute the response several times.

Parameters

StartEvent

Define the timer starting event (i.e. the row number in the Automation tag on which the event is defined). The parameter is optional. If no value is completed, the timer will be started automatically.

StopEvent

Define the timer stopping event (i.e. the row number in the Automation tag on which the event is defined). When StopEvent is executed, the timer will stop and will be restarted by Event only. This parameter is optional.

Period

Define the timer period.

Example of valid values:

10 – 10 seconds (units are unnecessary)
10s – 10 seconds
100ms – 100 milliseconds

Count

Define the count of repetitions. The parameter is optional and the default value is **0**, which means that the count of timer generated events is unlimited. Value 1 makes the timer behave as a Delay.

Variables

This block does not define any specific variables.

Example

Event generated three times in 1s intervals after the rise of event on row 1:

Event.Timer: StartEvent=1; Period=1s; Count=3

Event.HttpTrigger

The **HttpTrigger** block defines the event generated by receiving of an HTTP command from the intercom HTTP server. When the HTTP command http://ip_addr/enu/trigger/id is received, an event will be generated whose ID matches the value that follows 'trigger/' in the HTTP command. The intercom sends a simple reply to this request (200 OK).

Parameters

Name

Define a unique HTTP command identifier including alphabetical characters and digits.

Variables

The HttpTrigger event is always generated by the HTTP command which can carry a list of user parameters as included in the URI command.

http://ip_adresa/trigger/id?param1=value1&param2=value2

The list of parameters follows the ? character. Each parameter must include the name and value separated with the = character. If the list includes more parameters than one, & is used as the separator.

The HTTP-received parameters are available as HttpTrigger block variables. The variable name equals to the name of the parameter transferred.

Example

Event generated by receiving of the following HTTP command: http://ip_addr/enu/trigger/opendoor:

Event.HttpTrigger: Name=opendoor

Event.MulticastTrigger

The **MulticastTrigger** block defines the event generated by receiving of a command sent via **SendMulticastRequest**. The request is a message sent by UDP to a multicast address (235.255.255.250:4433) and can thus be received by multiple devices at the same time. The message includes the command ID (Command parameter) and additional optional parameters. The message can be password-secured (Password parameter).

Parameters

Command

Define the command ID to distinguish separate command types. The MulticastTrigger block responds to the SendMulticastRequest action only if the command identifier is the same. Any text containing the A-Z, a-z and 0-9 characters can be used for identification. The Upper/Lower case must be respected in the command name.

Password

Define the password to secure the command against unauthorised access. The password must match the value defined in the SendMulticastRequest action to which MulticastTrigger is expected to respond.

CheckTime

Enable/disable the check of the command receiving time against the time value included in the command message to eliminate attacks caused by repeating of an already processed message. Synchronised time (via the NTP server) on all command sending and receiving devices is required for this function.

Valid values:

0 – message time is not checked
1 – message time is checked (enhanced security)

The parameter is optional, the default value is **0**.

Variables

The MulticastTrigger event is generated whenever a mass command including the list of user parameters (Params parameter, MulticastRequest action) is received. Each of the parameters has a user-defined unique name and is available as a variable of the same name in the MulticastTrigger block.

Example:

Suppose a mass command generated by the MulticastRequest action is received, in which Params="AAA=123" is included. The MulticastTrigger event which

processes this command will automatically include value 123 for the AAA variable. This variable can be referred to in the interconnected blocks.

Example

Event generated by receiving of a mass opendoor command:

Event.MulticastTrigger: Command=opendoor

Event.AudioLoopTest

The **AudioLoopTest** block defines the event generated after the loudspeaker and microphone test (Audio Loop Test) is performed. The subsequent actions are executed based on the test result.

Parameters

Result

This parameter specifies the required test result.

Valid values:

any – the event is generated whenever the test is performed (regardless of the result).
passed – the event is generated whenever the test is successful.
failed – the event is generated whenever the test fails.

The parameter is optional, the default value is **failed**.

Variables

This block does not define any specific variables.

Example

An event generated after the audio loop test if the test result is negative (i.e. the microphone or loudspeaker is out of order):

Event.AudioLoopTest: Result=failed

1.4 Actions

2N® Helios IP Automation defines the following types of actions:

ActivateSwitch	switch activation
SetOutput	digital output state setting
BeginCall	outgoing call setup
AnswerCall	incoming call answer
EndCall	call termination
SendHttpRequest	HTTP command sending
SendMulticastRequest	command sending to multiple devices
PlayUserSound	user sound playing
StartMulticastSend	audio stream sending start
StopMulticastSend	audio stream sending stop
StartMulticastRecv	audio stream receiving start
StopMulticastRecv	audio stream receiving stop
SetCameraInput	external camera input selection

Action.ActivateSwitch

The **ActivateSwitch** block defines the action necessary for activation of the intercom switch as configured in the Switch 1 – 4 tags. The activity to be performed depends fully on the particular switch settings (digital output activation, HTTP command sending, etc.). Switch deactivation is controlled by the switch settings too.

Parameters

Event

Define the event to launch the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Switch

Define the switch to be activated (1 to 4).

Example

Activate switch 1 if the event defined on row 2 arises and the condition defined on row 3 is met:

Action.ActivateSwitch: Switch=1; Event=2; Condition=3

Action.SetOutput

The **SetOutput** block defines the action necessary for setting of the intercom output to the required level.

Parameters

Event

Define the event that launches the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Output

Define the output to be set.

Valid values:

relay1 – relay 1 on basic unit
relay2 – relay 2 on basic unit
output1 – output 1 on basic unit
output2 – output 2 on basic unit
cr_relay1 – relay 1 on card reader
cr_relay2 – relay 2 on card reader
cr_output – output 1 on card reader
redled - red LED indicator
led1 – LED 1 indicator
led2 – LED 2 indicator
led3 – LED 3 indicator

There may be different lists of valid values for different 2N® Helios IP models; refer to the Available Digital Inputs and Outputs subsection.

Level

Define the required output level. This parameter is optional.

Valid values:

0 – output deactivation
1 – output activation (default value)

Example

Activate Output1 if the event defined on row 2 arises:

Action.SetOutput: Output=output1; Event=2

Action.BeginCall

The **BeginCall** block defines the action necessary for establishing of an outgoing call to the defined telephone number, SIP URI or user number included in the intercom telephone directory.

Parameters

Event

Define the event to launch the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Number

Define the telephone number to be called.

Uri

Define the SIP URI to be called: sip:user@domain

User

Define the user number from the telephone directory to be called. The valid values are 1 through 999 (depending on the intercom model).

Enter just one of the above mentioned parameters (Number, Uri or User).

Example

An outgoing call will be established if the event defined on row 2 arises:

Action.BeginCall: Number=1001; Event=2

Action.AnswerCall

The **AnswerCall** block defines the action necessary for answering of an incoming call. In case no call is coming or the incoming call is not ringing, the action will not initiate any activity.

Parameters

Event

Define the event to launch the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Example

A call will be answered if the event defined on row 2 arises:

Action.AnswerCall: Event=2

Action.EndCall

The **EndCall** block defines the action necessary for termination of the currently made call. In case there is no active call via the intercom, the action will not initiate any activity.

Parameters

Event

Define the event to launch the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Example

A call is terminated if the event defined on row 2 arises:

Action.EndCall: Event=2

Action.SendHttpRequest

The **SendHttpRequest** block defines the action necessary for sending of an HTTP command to another LAN device. The HTTP command helps you control other devices in the LAN (IP relay, recording system, another intercom, etc.).

Parameters

Event

Define the event to launch the action.

Condition

Define the condition to be met to execute the action. This parameter is optional.

Uri

Define the standard HTTP URI including the destination address and, optionally, the path and other parameters.

Example

Send an HTTP command to the device with the IP address 192.168.1.1 if the event defined on row 2 arises:

Action.SendHttpRequest: uri=http://192.168.1.1/message; Event=2

Action.SendMulticastRequest

The **SendMulticastRequest** block defines the user command sending action to multiple devices. The sent command can be processed by the **MulticastTrigger** block. The command is a message sent by UDP to a multicast address (235.255.255.250:4433) and can thus be received by multiple devices at the same time. The message includes the command ID (Command parameter) and additional optional parameters (Params parameters). The message can be password-secured (Password parameter).

Parameters

Event

Define the event to execute this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Command

Define the command identifier to distinguish separate command types. The MulticastTrigger block responds to the SendMulticastRequest action only if the command identifier is the same. Any text containing the A-Z, a-z and 0-9 characters can be used for identification.

Params

Define one or more (comma-separated) command parameters to be included in the UDP message. Keep the "parameter_name=parameter_value" format.

Example:

Params="Address=192.168.1.1", "Port=10000"

The so-sent parameters will be available in the HttpTrigger event responding to this command as the Address and Port variables and can be used in HttpTrigger-tied actions, for example.

Password

Define the password to secure the command against unauthorised access. The parameter is optional. If no password is completed, the command is not secured. Use any text containing the A-Z, a-z and 0-9 characters.

Example

Send the opendoor command to all devices with the properly set Event.MulticastTrigger block in the network.

Action.SendMulticastRequest: Command=opendoor

Action.PlayUserSound

The **PlayUserSound** block defines the user sound playing action.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Sound

Select the user sound number (1 – 10).

Example

Play user sound 1:

Action.PlayUserSound: Sound=1

Action.StartMulticastSend

The **StartMulticastSend** block defines the starting action for audio stream sending to a multicast IP address. You can control up to four independent transmission channels. The RTP/UDP protocol is used and the data are in the PCMU format.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Channel

Define the channel number (1-4) to be controlled.

Address

Define the audio stream multicast IP address.

Port

Define the UDP port to which audio stream shall be sent.

Example

Start audio stream sending via channel 1 to address 239.0.0.1:10000:

Action.StartMulticastSend: Channel=1; Address=239.0.0.1; Port=1000

Action.StopMulticastSend

The **StopMulticastSend** block defines the stopping action for audio stream sending to a multicast IP address.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Channel

Define the channel number (1-4) to be controlled.

Example

Stop audio stream sending via channel 1:

Action.StopMulticastSend: Channel=1

Action.StartMulticastRecv

The **StartMulticastRecv** block defines the starting action for audio stream receiving and playing. You can control up to four independent transmission channels. The RTP/UDP protocol is used and the data are in the PCMU format.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Channel

Define the channel number (1-4) to be controlled.

Address

Define the audio stream multicast IP address.

Port

Define the UDP port to which audio stream shall be received.

Volume

Define the relative volume level for the audio stream to be played (from -6dB to +6dB).

Valid values:

-6 – minimum level
0 – mean level (default value)
6 – maximum level

The parameter is optional, the default value is **0**.

Example

Start audio stream receiving to multicast IP address 239.0.0.1:10000 via channel 1:

Action.StartMulticastRecv: Chanel=1; Address=239.0.0.1; Port=10000

Action.StopMulticastRecv

The **StopMulticastRecv** block defines the stopping action for audio stream receiving to a multicast IP address.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Channel

Define the channel number (1-4) to be controlled.

Example

Stop audio stream receiving via channel 1:

Action.StopMulticastRecv: Channel=1

Action.SetCameraInput

The **SetCameraInput** block defines the action that switches the video signal sources – either the integrated video camera or an external IP camera, or two inputs for analogue camera connection to the 2N® Helios IP Video Kit.

Parameters

Event

Define the event to launch this action.

Condition

Define the condition to be met for the action to be executed. This parameter is optional.

Type

Define the video signal type.

Valid values:

internal – internal camera (or external analogue video camera connected directly to the device)
external – external IP camera

The parameter is optional, the default value is **internal**.

Id

Define the video signal channel. The parameter is available with the 2N® Helios IP Video Kit model only and applicable only if the Type parameter is set to **internal**.

Valid values:

1 – analogue camera connected to input 1
2 – analogue camera connected to input 2

The parameter is optional, the default value is **1**.

Example

The video signal source is switched to external analogue camera input:

```
Action.SetCameraInput: Type=internal; Id=2
```

1.5 Conditions

2N® Helios IP Automation defines the following types of conditions:

ProfileState	time profile state
CallState	current call state
InputState	digital input state
LogicalAnd	logical AND of conditions
LogicalOr	logical OR of conditions
LogicalNot	condition negation
FlipFlopD	D-type flip-flop
FlipFlopRS	RS-type flip-flop
True	always true condition
False	always false condition

See below for details on the conditions and their parameters and use.

Condition.ProfileState

The **ProfileState** block defines the condition to be met in the case of active/inactive time profile.

Parameters

Profile

Define the time profile number (1 – 20 depending on the intercom model).

State

Define the required profile state. This parameter is optional.

Valid values:

active – active profile (default value)
inactive – inactive profile

Example

The condition is met for inactive time profile 1:

Condition.ProfileState: Id=1; State=Inactive

Condition.CallState

The **CallState** block defines the condition to be met in the case of a defined state of the currently made call.

Parameters

State

Define the call state.

Valid values:

idle – call is not being made
ringing – ringing in progress
connected – call connected

Direction

Define the call direction.

Valid values:

incoming – incoming calls
outgoing – outgoing calls
any – both directions

The parameter is optional, the default value is **any**.

Example

The condition is met for an inactive call:

Condition.CallState: State=Inactive

Condition.InputState

The **InputState** block defines the condition to be met in case the defined logic level gets connected to the defined digital input.

Parameters

Input

Define the digital input.

Valid values:

tamper – tamper switch
input1 – digital input 1
input2 – digital input 2
cr_input1 – digital input 1 on card reader
cr_input2 – digital input 2 on card reader

There may be different lists of valid values for different 2N® Helios IP models; refer to the Available Digital Inputs and Outputs subsection.

Level

Define the required digital input level. The parameter is optional.

Valid values:

0 – logic 0
1 – logic 1 (default value)

Example

The condition is met for an activated tamper switch (device not open):

Condition.InputState: Input1=tamper; Level=0

Condition.LogicalAnd

The **LogicalAnd** block helps you create groups of conditions. The block is fulfilled if all the conditions in the defined group are met.

Parameters

Condition

Define the list of conditions to be met. Separate the conditions with a comma.

Example

The condition is met if conditions 1, 2 and 3 are met at the same time:

Condition.LogicalAnd: Condition=1, 2, 3

Condition.LogicalOr

The **LogicalOr** block helps you create groups of conditions. The block is fulfilled if one condition at least of the defined group is met.

Parameters

Condition

Define the list of conditions to be met. Separate the conditions with a comma.

Example

The condition is met if conditions 1, 2 or 3 are met:

Condition.LogicalOr: Condition=1, 2, 3

Condition.LogicalNot

The **LogicalNot** block defines the condition to be met in case another defined condition is not met.

Parameters

Condition

Define the condition not to be met.

Example

The condition is met in case condition 1 is not met:

Condition.LogicalNot: Condition=1

Condition.True

The **True** block defines the condition to be met each time.

Parameters

There are no parameters in the True block.

Example

The condition will always be met:

Condition.True

Condition.False

The **False** block defines the condition not to be met each time.

Parameters

There are no parameters in the False block.

Example

The condition will always not be met.

Condition.False

Condition.FlipFlopD

The **FlipFlopD** block is a one-bit memory cell (variable), which records the state of another condition at the moment of rise of the defined event for later use. The FlipFlopD output can be used as a condition for control of actions in rather complex 2N® Helios IP Automation applications. It is a simulation of a D-type flip-flop circuit.

Parameters

ClockEvent

Define the event at which the current state of the condition is to be recorded.

Condition

Define the condition to be recorded at the rise of the ClockEvent.

ResetValue

Set the condition default value upon restart. The parameter is optional.
Valid values:

0 – condition is not met (default value)
1 – condition is met

Example

The state of the condition will be identical to the state of condition 2 at the rise of event 1:

Condition.FlipFlopD: ClockEvent=1; Condition=2

Condition.FlipFlopRS

The **FlipFlopRS** block is a one-bit memory cell (variable), whose state changes to 1 or 0 at the rise of defined events. The FlipFlopRS output can be used as a condition for control of actions in rather complex 2N® Helios IP Automation applications. It is a simulation of an RS-type flip-flop circuit.

Parameters

SetEvent

Define the event to set the condition into the 'met' state (1).

ResetEvent

Define the event to set the condition into the 'not met' state (0).

ResetValue

Set the condition default value upon restart. The parameter is optional.

Valid values:

0 – condition is not met (default value)
1 – condition is met

Example

The condition will be met at the rise of event 1 and not met at the rise of event 2:

Condition.FlipFlopRS: SetEvent=1; ResetEvent=2

1.6 Available Digital Inputs and Outputs

In this section, the digital inputs and outputs available on each 2N® Helios IP model are described.

2N Helios IP Vario

Outputs

relay1 – relay output on basic unit
relay2 – relay output on additional switch (if installed)
cr_relay1 – relay output 1 on card reader (if installed)
cr_relay2 – relay output 2 on card reader (if installed)
redled – red LED indicator under name tags (for no-display 9137xxxU models only)

Inputs

cr_input1 – digital input 1 on card reader (if installed)
cr_input2 – digital input 2 on card reader (if installed)

2N Helios IP Force/Safety

Outputs

relay1 – relay output on basic unit
output1 – digital 12V output on basic unit (for board versions 555v3 and higher, digital 12V output is connected with relay output 1 in 555v2 boards)
relay2 – relay output on additional switch (if installed)
output2 – digital 12V output on additional switch (if installed)
cr_relay1 – relay output on card reader (if installed)
cr_output1 – digital 12V output on card reader (if installed)
redled – red LED indicator on card reader (if installed)

Inputs

tamper – tamper switch (if installed)
cr_input1 – digit input 1 on card reader (if installed)
cr_input2 – digital input 2 on card reader (if installed)

2N Helios IP Uni

Outputs

relay1 – relay output on basic unit

Inputs

Not available.

2N Helios IP Audio/Video Kit

Outputs

relay1 – relay output
output1 – digital output 1
output2 – digital output 2
led1 – LED 1 control output
led2 – LED 2 control output
led3 – LED 3 control output

Inputs

input1 – digital input 1
input2 – digital input 2

1.7 Examples of Use

Calling to Dispatching Office in Case of Unauthorised Door Opening

Specification

Call the selected telephone number whenever the tamper switch is disconnected (device opened).

Block diagram

The rising edge on the tamper input (1: Event.InputChanged) initiates calling to the defined telephone number (2: Action.BeginCall).

Intercom settings

1: Event.InputChanged: Input=tamper
2: Action.BeginCall: Number=1111; Event=1

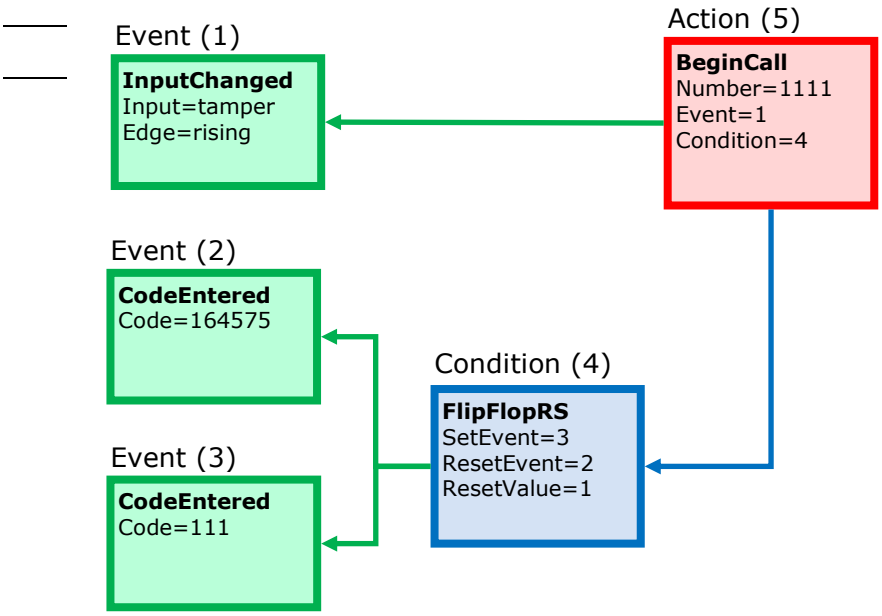
Calling to Dispatching Office in Case of Unauthorised Door Opening with Service Code Blocking Option

Specification

Call the selected telephone number whenever the tamper switch is disconnected (device opened). Enable blocking and re-enable numerical code alarm entered from the intercom keypad.

Block diagram

The rising edge on the tamper input (1: Event.InputChanged) initiates calling to the defined telephone number (5: Action.BeginCall) in case the defined condition is met. The condition (4: Condition.FlipFlopRS) is validated by the intercom restart or entering the selected code (2: Condition.CodeEntered) from the numerical keypad. If another code is entered (3: Condition.CodeEntered), the condition will be invalid.



Intercom settings

1: Event.InputChanged: Input=tamper; Edge=rising
2: Event.CodeEntered: Code=164575
3: Event.CodeEntered: Code=111
4: Condition.FlipFlopRS: SetEvent=3; ResetEvent=2; ResetValue=1
5: Action.BeginCall: Number=1111; Event=1; Condition=4

Door Opening by RFID Card

Specification

Activate the door contact switch by entering the proper RFID card.

Block diagram

Entering an RFID card with the defined ID (1: Event.CardEntered) activates switch 1 (2: Action.ActivateSwitch).

Intercom settings

1: Event.CardEntered: Card=0*0000
2: Action.ActivateSwitch: Switch=1; Event=1

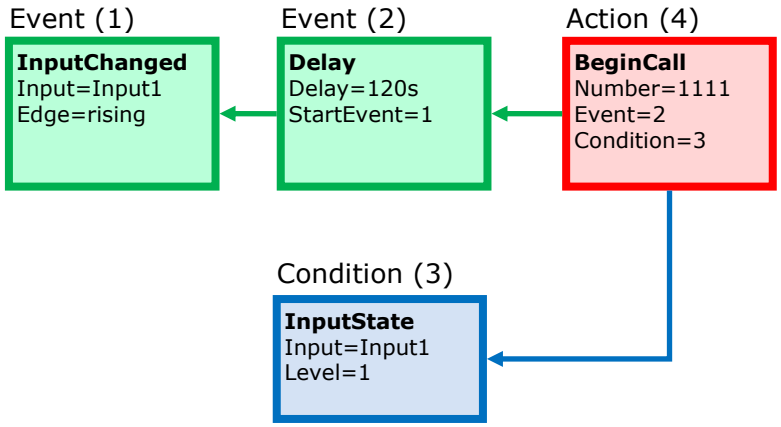
Alarm (Dispatching Office Call) Caused by Over 2-Min Long Door Opening

Specification

Call the dispatching office in case the door remains open for more than 2 minutes. It is supposed in the example that the door opening signalling contact is connected to Input1.

Block diagram

Whenever the door opens, the rising edge on Input1 signal (1: Event.InputChanged) calls the defined telephone number (4: Action.BeginCall) with a 120s delay (2: Event.Delay). The call is only executed if the door remains open for more than 120s (3: Condition.InputState).



Intercom settings

1: Event.InputChanged: Input=input1; Edge=rising
2: Event.Delay: Delay=120s; StartEvent=1
3: Condition.InputState: Input=input1; Level=1
4: Action.BeginCall: Number=1111; Event=2; Condition=3

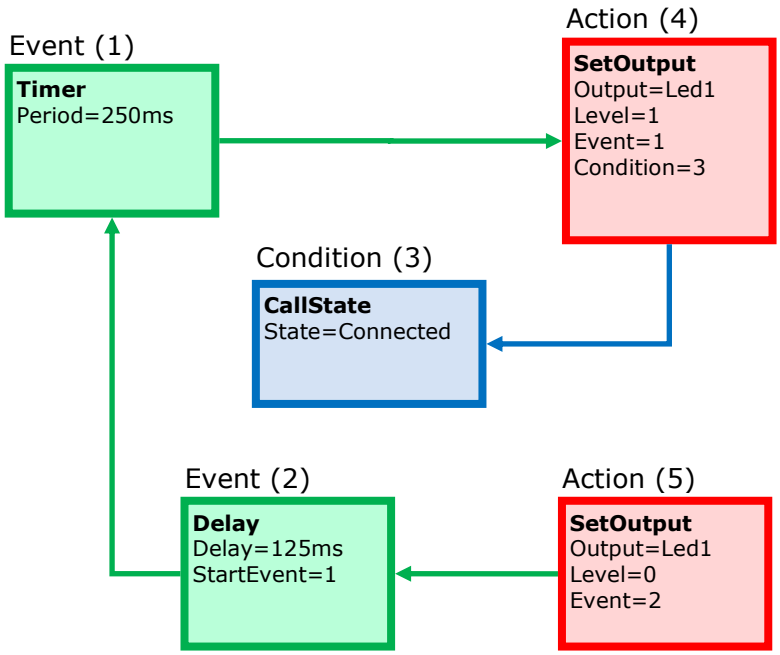
LED Flashing during Call / Electric Door Lock Opening

Specification

Enable LED flashing during an active call.

Block diagram

Enable LED flashing by a combination of the periodic timer (1: Event.Timer) and delay (2: Event.Delay). These two blocks define the period (250ms) and duty cycle of the signal or the LED shining period (125ms). These two events are tied with the on-switching (4: Action.SetOutput) and off-switching (5: Action.SetOutput) actions. The LED switch-on action is conditioned by the active call (3: Condition.CallState).



Intercom settings

1: Event.Timer: Period=250ms
2: Event.Delay: Delay=125ms; StartEvent=1
3: Condition.CallState: State=Connected
4: Action.SetOutput: Output=led1; Level=1; Event=1; Condition=3
5: Action.SetOutput: Output=led2; Level=0; Event=2

2N TELEKOMUNIKACE a.s.

Modřanská 621, 143 01 Praha 4, Česká Republika
Tel.: +420 261 301 111, Fax: +420 261 301 999
E-mail: obchod@2n.cz
Web: www.2n.cz