ibsysComponents

Documentation for the ibsysComponents module

Version: V2.0

20th April 2026

IBSYS GmbH
Lagerplatz 6, 8400 Winterthur
Author: Levi Jetzer

General

The ibsysComponents are function blocks that can be used for programming Niagara controllers. They mainly contain components for HVAC applications.

Starting with version 4.3.2.2, license credits were introduced as a license unit. All components for which no explicit number of license credits is specified require 1 license credit.

Compatibility

The ibsysComponents module can be used from Niagara version N4.10.

Version

This documentation applies from module version 4.4.1.2.

Contact

If you have any questions, comments, suggestions or error messages, please contact our technical support:

support@ibsys.ch

www.ibsys.ch

Version index

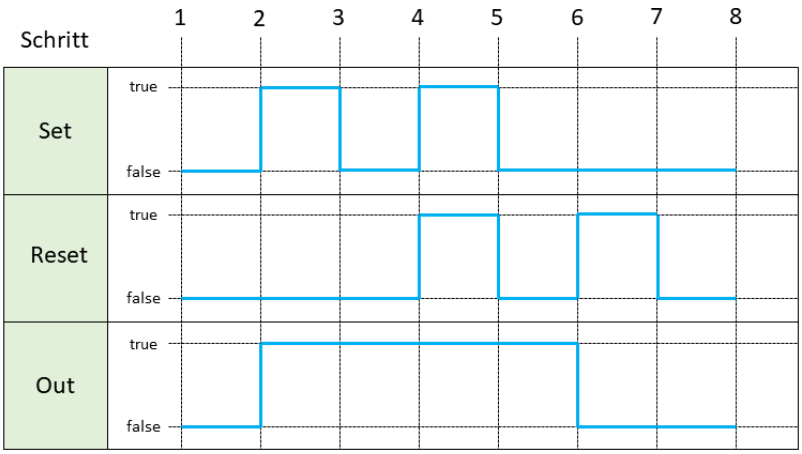
Version	Date	Comment	Author
V1.0	31/01/2024	Created	Levi Jetzer
V1.1	11/05/2024	Changed to module release V4.1.1.2	Levi Jetzer
V1.2	22/07/2024	Updated to module release V4.2.1.2; Add16, Min16, Max16, VavSetpointCalc and HistoryCsvExporter added	Levi Jetzer
V1.3	08/08/2024	Edited on module release V4.2.1.12, BooleanChanged, NumericChanged, EnumChanged, StringChanged	Levi Jetzer
V1.4	08/12/2025	Edited on module release V4.3.2.2, AlarmRoutingFilter & license credits	Levi Jetzer
V2.0	20/04/2026	Edited on module release V4.4.1.2, AlarmToString, MakeSimpleJson, GetJsonObject, HistoryCsvExporter, StringBase64Encoder	Levi Jetzer

Table of contents

1. FlipFlops

1.1. SR

The SR flip-flop is a bistable function block with dominant setting and is one of the standard IEC components. The function is as follows:

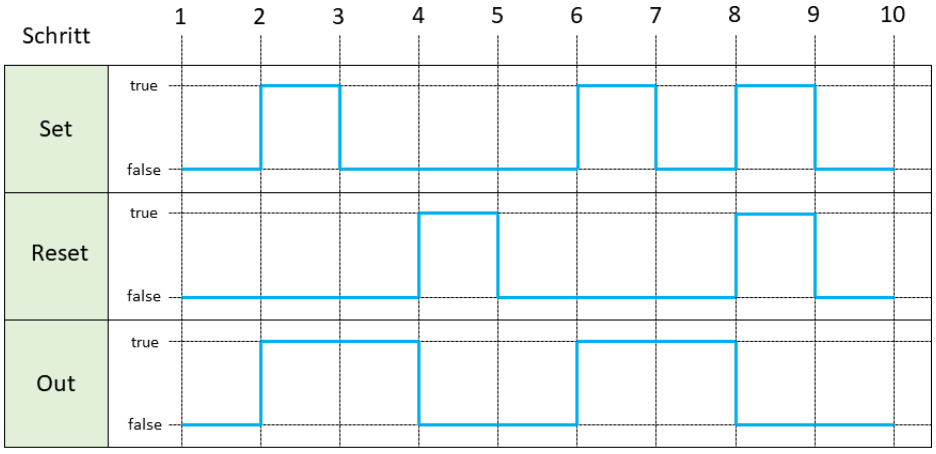


The value in the Out slot is saved in each case.

1.2. RS

The RS flip-flop is a bistable function block with dominant reset and is one of the standard IEC components. The function is as follows:

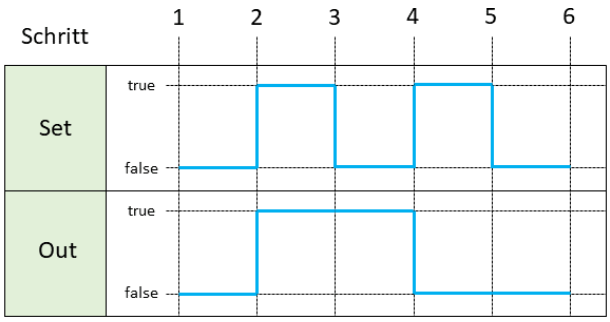
The value in the Out slot is saved in each case.



1.3. FlipFlop

The flip-flop is a bistable function block with one input. The function is as follows:

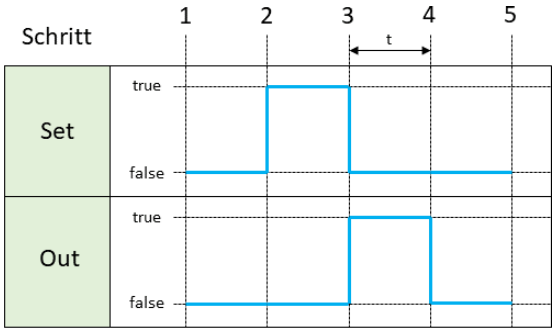
The value in the Out slot is saved in each case.



1.4. NegativeEdge

The NegativeEdge is a negative edge detection. The function is as follows:

The output is set to true after a negative edge for an adjustable time (t).



2. Logic

2.1. And16

The And16 is a standard AND component, but has 16 inputs. The difference to the AND from the kitControl module is the ‘Out Not’ slot, in which the negated value of the ‘Out’ slot is output.

2.2. Or16

The Or16 is a standard OR component, but has 16 inputs. The difference to the OR from the kitControl module is the ‘Out Not’ slot, in which the negated value of the ‘Out’ slot is output.

2.3. Average16

The Average16 is a standard AVERAGE component, like those in the kitControl module, but has 16 inputs.

2.4. Add16

The Add16 is an ordinary ADD component, like those in the kitControl module, but has 16 inputs.

2.5. Min16

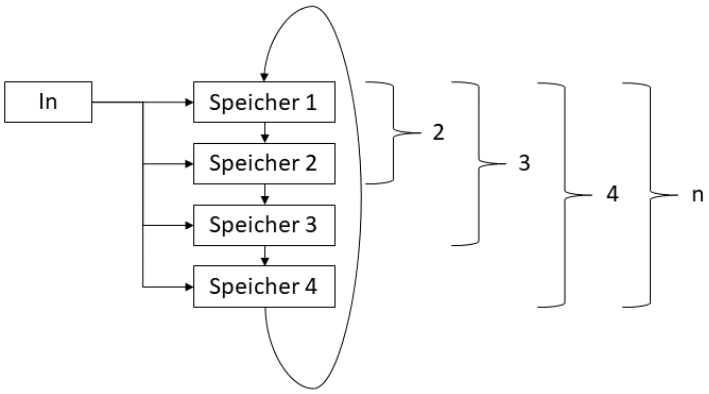
The Min16 is an ordinary MIN component, like those in the kitControl module, but has 16 inputs.

2.6. Max16

The Max16 is an ordinary MAX component, like those in the kitControl module, but has 16 inputs.

2.7. AverageValues

The AverageValues is a function block for creating an average of stored values. The update period specifies the interval at which the value is to be read in and saved at the In slot. If the update interval is 0, the average is calculated each time the input value changes. The Slot Number Of Values specifies how many values are to be saved. The average is only ever calculated using the stored values.



The output is recalculated and set after each interval.

2.8. BooleanLastChanged

The BooleanLastChange is a function block that has four In slots and one Out slot.

The function block outputs the value of the In slot that last changed at the Out slot.

‘Zero’ at an input is not counted as a value.

2.9. NumericLastChange

The NumericLastChange is a function block that has four In slots and one Out slot.

The function block outputs the value of the In slot that last changed at the Out slot.

‘Zero’ at an input is not counted as a value.

2.10. EnumLastChange

The EnumLastChange is a function block that has four In slots and one Out slot.

The function block outputs the value of the In slot that last changed at the Out slot.

‘Zero’ at an input is not counted as a value.

2.11. StringLastChange

The StringLastChange is a function block that has four In slots and one Out slot.

The function block outputs the value of the In slot that last changed at the Out slot.

‘Zero’ at an input is not counted as a value.

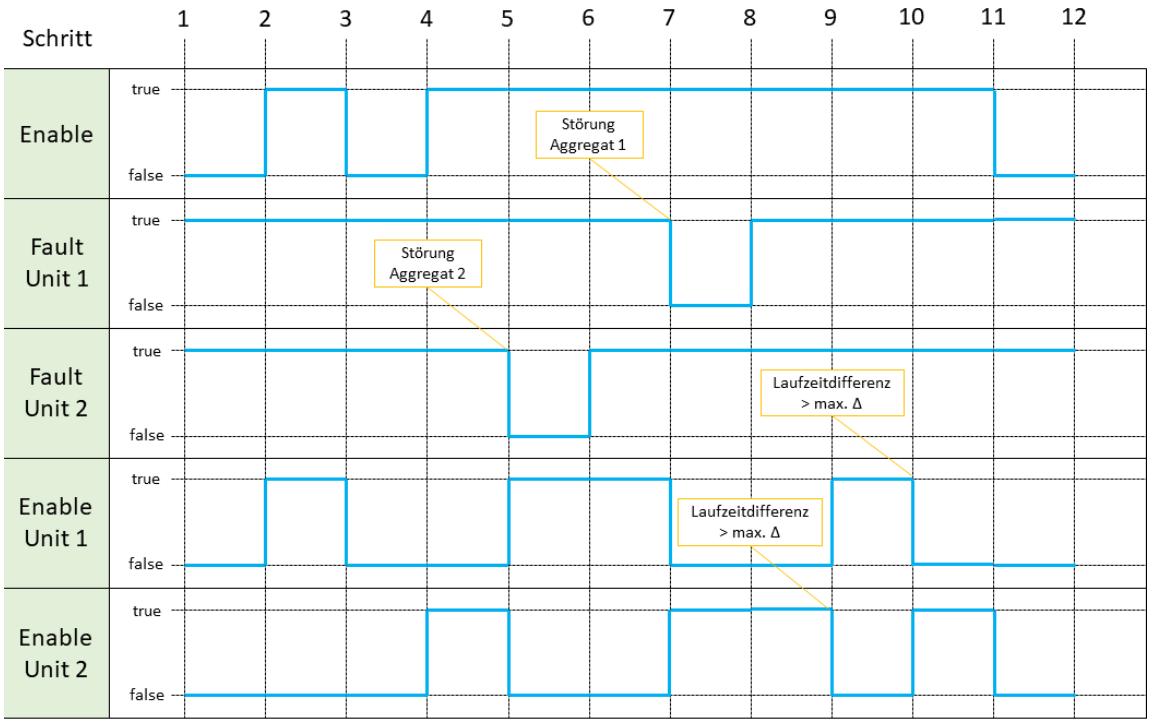
3. HVACLogic

3.1. DewPointCalculator

The dew point calculator calculates the absolute humidity and the dew point based on the temperature and relative humidity. The calculation is always carried out again when the value at the temperature input or the relative humidity input changes.

3.2. RuntimeSwitch

The RuntimeSwitch is a module for runtime and fault switching between two units. When enabled, the genset with the shorter runtime is always switched on first. If the difference between the runtime of the running genset and the runtime of the stationary genset is greater than the maximum delta, the gensets are switched over.



If a running genset has a fault, the second genset takes over, regardless of its running time. If a non-running genset has a fault, the runtime switchover is not carried out, even if the conditions for this would have been met. If both gensets have a fault, the Total Failure output is set to true.

3.3. OperatingMode

The OperatingMode module can average a temperature value over an adjustable time and then output an operating mode based on threshold values. For value averaging, 288 values are recorded, which corresponds to an interval of 5 minutes over 24 hours.

If the status ‘zero’ is detected at the outdoor temperature input, the values are no longer included in the average value calculation. This status is signalled at the output of the average outside temperature by the status ‘stale’. As soon as the value at the input no longer has the status ‘zero’, the average value is calculated correctly again and the status at the output changes back to ‘ok’.

The outputs are set as follows:

(average outside temperature > cooling limit On) & heating Off = cooling On

average outside temperature < cooling limit Off = cooling Off

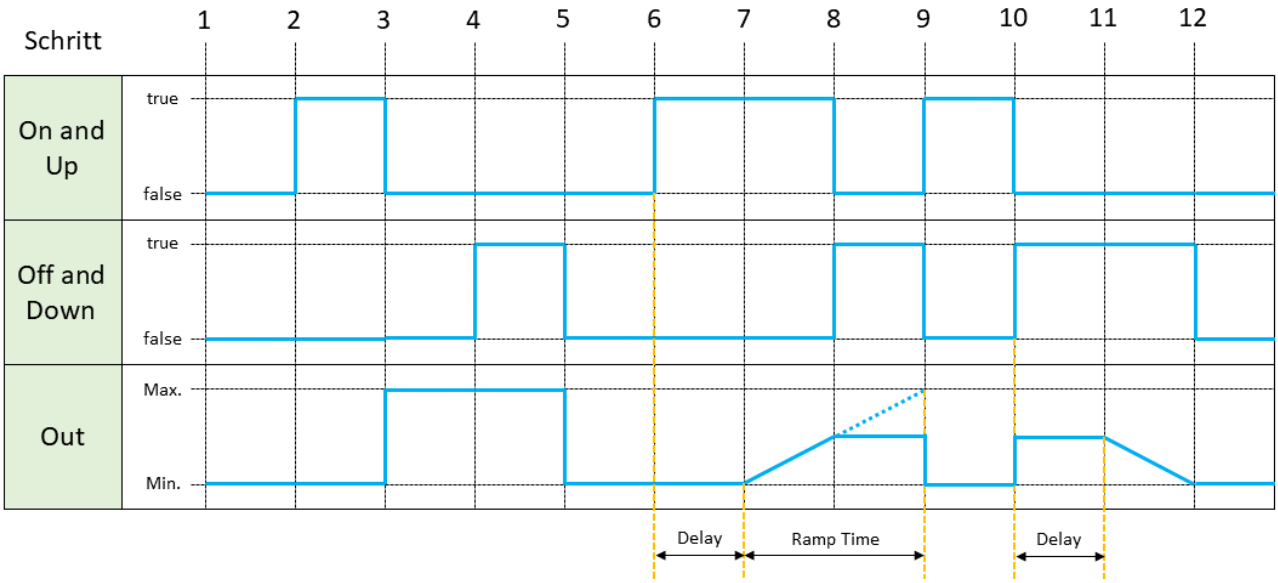
(average outside temperature < heating limit On) & cooling Off = heating On

average outside temperature > heating limit Off = heating Off

These comparisons automatically create a deadband, e.g. between Heating Off and Cooling On. This deadband was programmed in on purpose.

3.4. Dimmer

The dimmer is a component that can be used in lighting applications. The function is as follows:



Watch out: Recognising a digital input under 300ms is difficult with Niagara, as all inputs are read in at different speeds via bus protocols. For this reason, a certain amount of caution is required when evaluating the function of the dimmer.

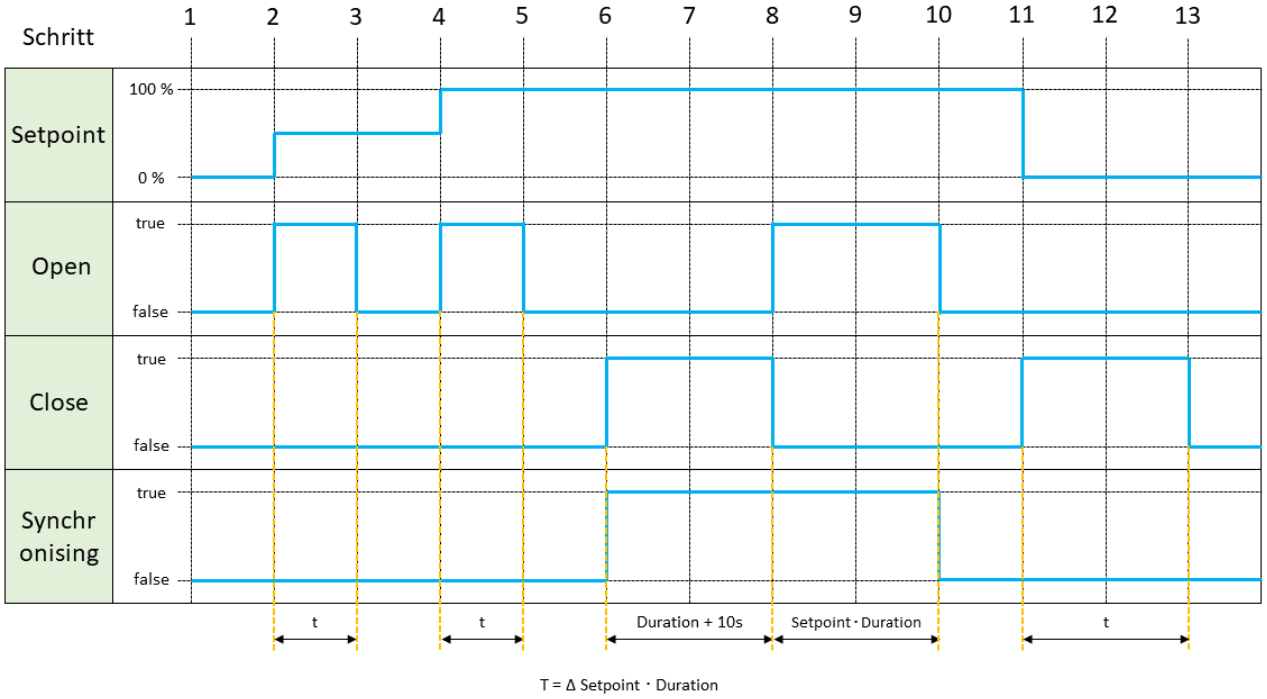
3.5. ThermalEnergyMeter

The ThermalEnergyMeter module can calculate the thermal energy in hydraulic applications. It can therefore take over the task of a missing meter. All that is required are the flow and return temperatures and the flow rate. The meter can be switched on or off via the 'Enabled' flag.

The density of the medium and the resulting heat capacity are calculated based on the measuring location and the temperature measured there (flow or return). The medium always refers to water. The 'GlycolContent' can be used to specify the glycol content if it is a cooling application. The glycol content has an influence on the density and heat capacity.

3.6. ThreePointDrive

The ThreePointDrive is a module for controlling three-point drives. The function is as follows:



If synchronisation is switched on, the drive is fully closed during the synchronisation interval (+10s safety) and then opened again according to the setpoint. This is used to compensate for the inaccuracy that accumulates over time.

3.7. SwitchingHysteresis

The SwitchingHysteresis function block represents a switching hysteresis. This is used, for example, to switch a heating or cooling mode on or off above or below a certain threshold value.

Slot Out is set to true if the value at slot inA minus the value of HysteresisOn is greater than the value of inB.

The Slot Out is set to false if the value at Slot inA plus the value of HysteresisOff is less than the value of inB.

3.8. PlantSwitch

The PlantSwitch function block represents a plant switch to which a hardware and software plant switch can be connected. The inputs for the software plant switch only have an effect on the outputs if the hardware plant switch inputs have the status zero or the hardware plant switch is set to Auto. If several inputs are present at the same time, Off always has priority.

3.9. VavSetpointCalc

The VavSetpointCalc can perform the calculation of shifted setpoints. This is used to reset the Vmin/Vmax values set in the VAV or to recalculate the relative setpoint in it. This means that the setting of Vmin or Vmax of the VAV can be outsourced to the module and ‘reset’ as required without access to the VAV. In addition, the VAV can also be used as a static pre-controller and regulated to a fixed volume setpoint. There is no need to measure the setpoint volume flows on site.

The ‘Relative Setpoint’ slot can be set using the link or via the ‘Set Rel Setpoint’ action. Based on the set Vmin/Vmax values (if Vmin and Vmax = 0, 0 and Vnom apply) and the ‘Volumetric Flow Setpoint Facets’ min and max values, the “Out” is then calculated as the relative setpoint.

The ‘Volumetric Flow Setpoint’ can be set using the link or via the ‘Set Volume Flow Setpoint’ action. The ‘Out’ is then also calculated as a relative setpoint based on the set parameters. The rules described above also apply here.

4. Converters

4.1. DecToHex

The DecToHex function block can convert decimal numbers to hexadecimal numbers. The input is numeric, the output is a string.

4.2. HexToDec

The HexToDec function block converts a hexadecimal input, formatted as a string, into a numerical decimal number.

4.3. IntBitsToFloat

In the IntBitsToFloat function block, integer input values are output as floats at the output.

4.4. BitsToNumericDemux

The BitsToNumericDemux function block can output up to 64 individual bits as a decimal number at the numeric output. This function block helps to write to ModBus registers, for example.

The number of desired bit inputs can be edited at the ‘NumberOfInputBits’ property.

4.5. VolumetricFlowToVolume

The VolumetricFlowToVolume function block can output and cumulate a flow rate during a period of time as a volume. This function can be helpful, for example, if no counter is available but the flow rate is known when the pump is switched on.

4.6. PowerToEnergy

The PowerToEnergy module can convert a pending power in kW to an energy value based on time and store and output it cumulatively.

4.7. StringToAsciiInt

The StringToAsciiInt function block can convert string characters as ASCII code into an integer. It always takes two characters, stores the first character in the first 8 bits of the integer and the second character in the second 8 bits.

4.8. AsciiIntToString

The AsciiIntToString function block can convert up to 64 ASCII code integer values into a string. The first 8 bits and the last 8 bits of an integer are always an ASCII character.

4.9. UnitConverter

The UnitConverter module can convert values between units. For example, between l/s and m3/h. However, only units with the same base can be converted.

5. Util

5.1. JSON

5.1.1. GetJsonObject

The GetJsonObject component can be used to extract a JSON object from a string based on a defined key. This component can, for example, be used to parse an API response and retrieve a specific value. The value is output as a string at the Out slot. However, thanks to the Niagara framework and its compatibility, a value such as ‘true’ can be linked to a BooleanWritable and is also interpreted as a Boolean value.

5.1.2. MakeSimpleJsonObject

The MakeSimpleJsonObject component can be used to create a JSON object from a Niagara data type, which is output via the Out slot. The UseEnumOrdinal slot allows you to choose whether the enum value is transferred to the JSON object as a string (tag) or as an integer (ordinal).

Note: A value is only included in the JSON object if its status is valid. If several values and statuses are valid at the same time, only the last one in the list from the top is included in the JSON object.

5.2. AlarmRoutingFilter

The AlarmRoutingFilter block can be used as a filter, for example, between an alarm class and an alarm receiver. The filter only forwards alarms that meet the filter criteria.

Slotname	Description	Type
Time Range	Time range in which alarms are forwarded	BTimeRange
Days Of Week	Weekdays on which alarms are forwarded	BDaysOfWeekBits
Transitions	Alarm transitions that are forwarded	BAlarmTransitionBits
Route Acks	Value to control the forwarding of acknowledgments	BBoolean
Alarm Filter	Filter to control the forwarding of alarms	BAlarmFilter

Alarm filter entries can be activated or deactivated using a flag. This disables the filter, for example, without changing its settings.

5.2.1. Alarm Properties

Key	Description	Type	Example Value
timestamp	Timestamp	AbsTime	20.04.2026 17:00
uuid	Alarm UUID	String	0000-0000…
sourceState	Source state	String	offnormal
ackState	Ack state	String	unacked
ackRequired	Ack required	Boolean	true
source	Source	String	
alarmClass	Alarm class	String	defaultAlarmClass
priority	Priority	Integer	0…255
normalTime	Normal time	AbsTime	20.04.2026 17:00
ackTime	Ack time	AbsTime	20.04.2026 17:00
user	User	String	Unknown User
alarmData	See Alarm Data	String	
alarmValue	Alarm value	Boolean, Double, DynamicEnum etc.	true/50,0
lastUpdate	Last update	AbsTime	20.04.2026 17:00

5.2.2. Alarm Data

The alarm data is a special property that can be freely configured in the filter. However, the alarm data of an alarm contains a great deal of information, which is why the most important keys are listed below.

Key	Description	Type	Example Value
alarmValue	alarm value	Boolean, Double, DynamicEnum etc.	true/50,0
Count	alarm counter	Integer	42
deadband	dead band	String	1,0
escalated	Escalated	String	level1
faultValue	fault value	String	50,0
fromState	from state	String	offnormal
highLimit	high limit	String	50,0
lowLimit	low limit	String	-20,0
msgText	message text	String	OK
notes	notes	String	Note
numericValue	numeric value	Integer	0
offnormalValue	offnormal value	String	true
presentValue	present value	String	false
sourceName	source name	String	BooleanWritable
status	status	String	{overridden} @ 1
timeDelay	time delay	RelTime	0 ms
timeDelayToNormal	time delay to state «normal»	RelTime	0 ms
TimeZone	timezone	TimeZone	Europe/Berlin (+1/+2)
toState	to state	String	normal

Furthermore, custom key-value pairs can be added to an AlarmExtension, which will then appear in the alarm data. These key-value pairs are called “meta data” in the AlarmExtension. The keys added to the meta data can then be used to filter the alarm data.

When entering filters, it is important that the key corresponds exactly to the spelling above. It may appear to be an error that, for example, Count is written with a capital letter, but this is not the case. The spelling of the keys in the table above is correct.

Furthermore, numerical values that are added with a string filter must be separated with a “,” as a separator.

The best way to see how the individual values should be written is to open an alarm and view the alarm data. The spelling must match exactly.

5.3. AlarmToString

The AlarmToString component can be used to convert alarms into a string. The alarm is converted into a JSON-formatted string at the time it occurs and output via the Out slot.

The JSON keys are based on the valid alarm properties. These can be found in the Alarm Properties chapter. The alarm data is output in the JSON object ‘alarmData’ with the specified keys.

Note: All timestamps are converted to Unix seconds and written to the JSON as integers. This ensures interoperability.


```
{
  "ackTime":0,
  "alarmClass":"Prio_1",
  "alarmData":{
    "fromState":"normal",
    "escalated":"",
    "Count":1,
    "presentValue":"60,0",
    "toState":"highLimit",
    "TimeZone":"Europe/Berlin (+1/+2)",
    "alarmValue":60,
    "offnormalValue":"60,0",
    "msgText":"",
    "deadband":"0,0",
    "timeDelay":"0 ms",
    "company":"IBSYS",
    "location":"Winterthur",
    "sourceName":"NumericWritable",
    "highLimit":"50,0",
    "timeDelayToNormal":"0 ms",
    "status":"{ok} @ def"
  },
  "source":"local:|station:|slot:/Services/AlarmService/NumericWritable/OutOfRangeAlarmExt",
  "priority":255,
  "uuid":"874bdf45-d714-44e6-ad71-79697bec871e",
  "ackState":"unacked",
  "normalTime":0,
  "alarmValue":60,
  "lastUpdate":1776697570,
  "ackRequired":true,
  "sourceState":"offnormal",
  "user":"Unknown User",
  "timestamp":1776697570
}
```


5.4. StringReplace

The StringReplace function block receives a BStatusString at the ‘In’ slot, searches for string characters in the ‘Search’ slot and replaces these with the characters in the ‘Replace’ slot. It then outputs the new string at the ‘Out’ slot.

5.5. HistoryCsvExporter

With the HistoryCsvExporter component, histories can be exported to a .csv file. The slots and their functions are described below:

Slotname	Description	Type
Status	Status of the component	BStatus
Fault Cause	Reason for error in the event of an error	String
Make File	If this option is disabled, the file is only output to the “File” topic slot and no file is generated	Boolean
Directory	Path under which the export is stored	Ord
File Name	File name of the .csv file	Format
Append Date Time	Appending the current timestamp to the file name	Boolean
Append Date Time Format	Formatting of the current timestamp which is appended to the file name	String
Target	Resulting path and file name of the export	String
Trigger	Trigger for the export	TimeTrigger
Last Export	Time of the last successful export	BAbsTime
Include Header	Include column titles in the export	Boolean
Export Props	Properties to be exported to the file	HistoryCsvExportStructure
History Ords	Histories from which the data is to be exported	OrdList
Time Range	Time range to be included in the export	DynamicTimeRange
Date Format	Date format for the date column in the export	String
Time Format	Time format for the time column in the export	String
Date Time Format	Time format for the timestamp column in the export	String
Value Format	Value format for the value column in the export	String
Csv Delimiter	.csv separator	String
File	File output for sending via email (ibsysAddOns)	Topic(Blob)

5.5.1. Properties

With four exceptions, the exported properties are history properties.

History Properties:

- History Name
- Date
- Date Time
- Date Time
- Value
- Unit
- Trend Flags
- Status

Parent Component Properties:

- User Property 1
- User Property 2
- User Property 3
- User Property 4

These properties refer to a property slot on the parent of the history extension.

In order for the properties to be recognised on the parent, they must be named exactly as follows: ‘userProperty1’, ‘userProperty2’, ‘userProperty3’ or ‘userProperty4’.

The displayName of the properties may differ for better labelling. This makes it possible to include additional properties in the export, which are available as static information on the point.

The column names can be freely overwritten; the property name is used as the default. This is done by making an entry in the string input field (Column Name) next to the property.

Date Time Format

The timestamp can be freely formatted. The following representations apply:

YYYY: Year in four digits (e.g. 2024)
MM: Month in two digits (e.g. 07 for July)
MMM: Month in long notation (e.g. July)
DD: Day in two digits (e.g. 22)
HH: Hours in 24h format in two digits (e.g. 16 for 4pm)
hh: hours in 12h format in two digits (e.g. 4 for 4pm)
mm: Minutes in two digits
ss: Seconds are not displayed in the date as a file name attachment

An underscore is recommended as a separator; spaces and a few other characters are not accepted as they violate the file name convention.

Important: The content of the user properties is written to the export for all rows at the time of export.

Note: If there are several trend flags or statuses, these are separated from each other with a comma. It is recommended to use a semicolon as a delimiter or not to export the trend flags and the status.

Note Topic(Blob): In the topic slot, the CSV file is passed on solely as a byte array (blob); the file format, filename, etc. have no bearing on this and are not included in it.

5.6. BooleanChanged

The BooleanChanged function block outputs a true at the output for a configurable time in the event of a change at the input. A status change is not evaluated as a value change.

5.7. NumericChanged

The NumericChanged function block outputs a true at the output if there is a change at the input for an adjustable time. A status change is not evaluated as a value change.

5.8. EnumChanged

The EnumChanged function block outputs a true at the output if there is a change at the input for an adjustable time. A status change is not evaluated as a value change.

5.9. StringChanged

The StringChanged function block outputs a true at the output if there is a change at the input for an adjustable time. A status change is not evaluated as a value change.

5.10. StringBase64Encoder

The StringBase64Encoder block can Base64-encode a string received at the input slot and output it via the output slot. The input is first converted to UTF-8 bytes and then Base64-encoded.

6. License Credits

Unless otherwise specified here, each instance of a component requires one license credit.

Component	Required license credits (per instance)
DewPointCalculator	100
RuntimeSwitch	100
ThermalEnergyMeter	100
AlarmRoutingFilter	500
HistoryCsvExporter	10,000