KIS.ME

Keep it simple. Manage everything.

KIS.MANAGER

KPI-Manual

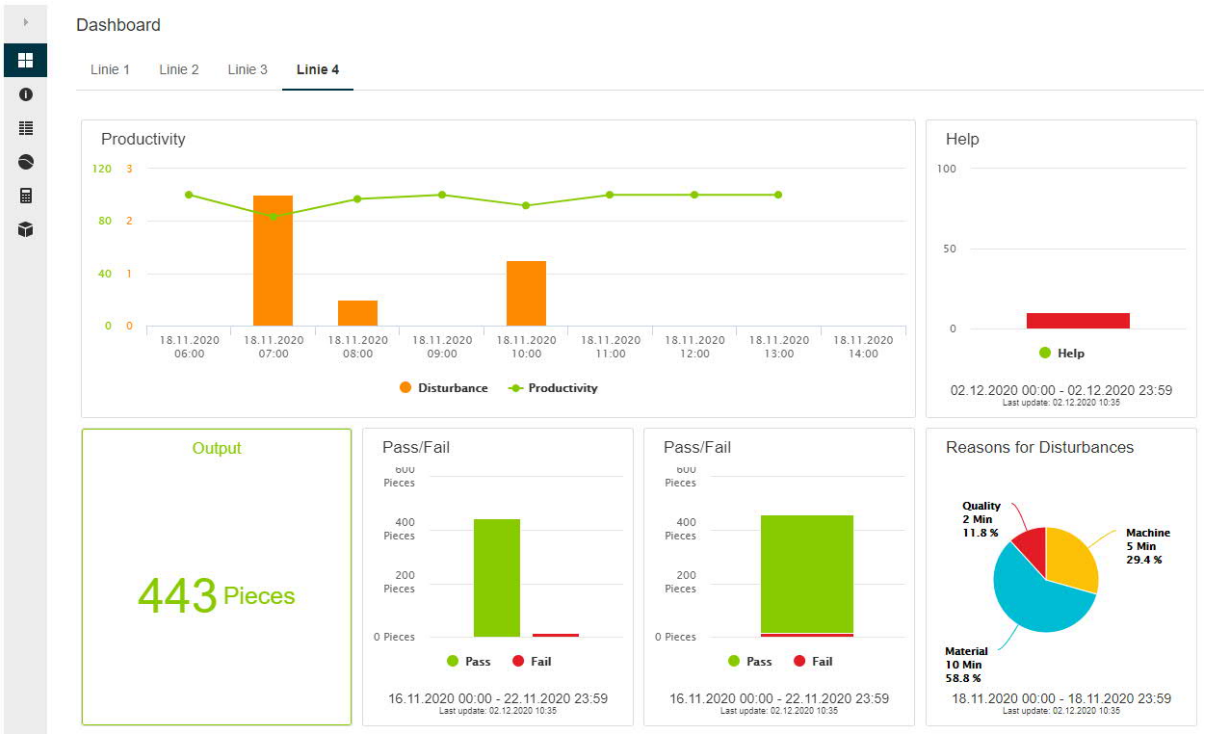
Version 1.1, 07/2021

Inhaltsverzeichnis

1 Introduction

Key figures are a key pillar for analyzing and evaluating processes. They provide the necessary transparency and allow targeted actions.
The KIS.MANAGER provides attractive options for this. The KIS.Devices (KIS.BOX, KIS.LIGHT, etc.) collect data locally, and the data is processed and presented in the KIS.MANAGER.

The result can look e.g. as follows:



Different widgets are available for presenting key figures on the various dashboards of the KIS.MANAGER.

The following manual explains the route from datapoint to key figure and also includes the comprehensive presentation options.

2 Datapoints

The datapoints form the input for every key figure calculation.

A datapoint is an exchanged value between KIS.Device and KIS.MANAGER. A distinction is made between:

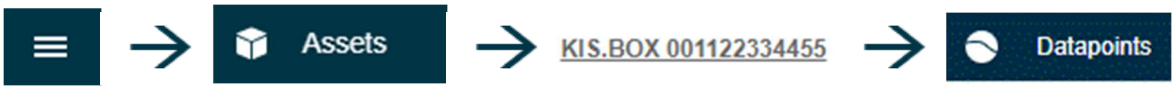
- ➔ Condition data (button pressed, LED color green, INPUT high,…)
- ➔ Metadata (firmware version, hardware version, WLAN signal strength,…)

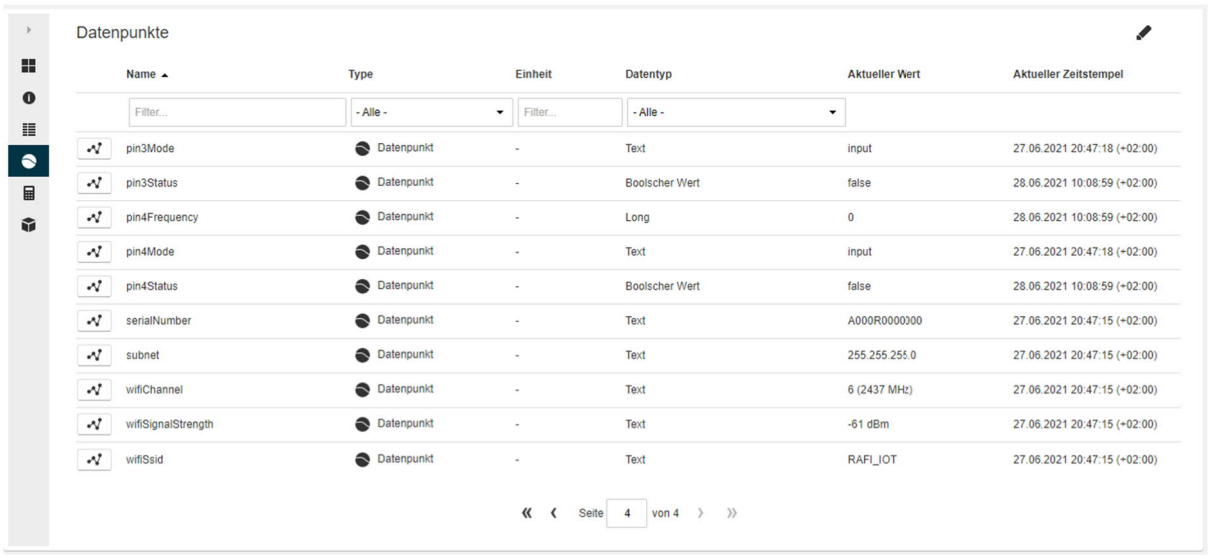
The complete communication between KIS.Device and KIS.MANAGER also takes place via the datapoints.

Condition data is relevant for calculating key figures. It transports the process information which is to be processed via the key figures.

2.1 Datapoints as the basis for key figures

For each KIS.Device, the KIS.MANAGER provides a complete overview regarding the datapoints exchanged:





Basically, you can evaluate the following types of datapoints:

- ➔ Double - Floating-point number (e.g.: 1.234)
- ➔ Long - Whole number (e.g.: 1)
- ➔ Boolean - Logical value (e.g.: true)

An overview of the datapoints which can be evaluated can be found in the appendix in Chapter 8.1

2.2 Datapoints as a labor-saving approach

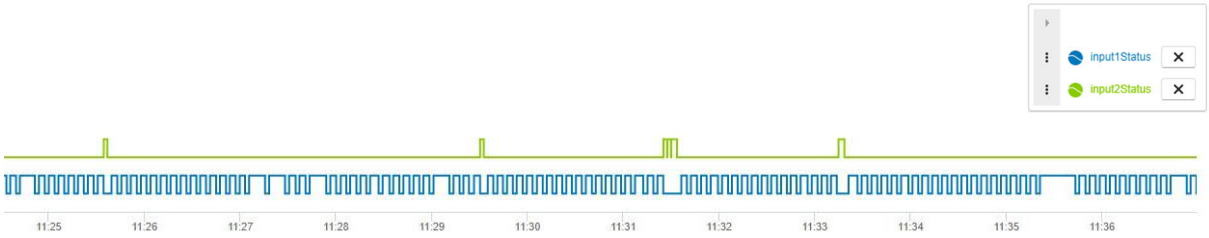
The overview of the datapoints of an asset makes it easier to create KPIs and check whether the calculated result is plausible.

The overview provides the following options:

- ➔ Filtering according to datapoints which can be evaluated
- ➔ The current value of one/multiple datapoints can be viewed
- ➔ The profile over time of one/multiple datapoints can be viewed
- ➔ The historical values of one/multiple datapoints can be exported
- ➔ The history of a datapoint can be deleted (deletion period adjustable)

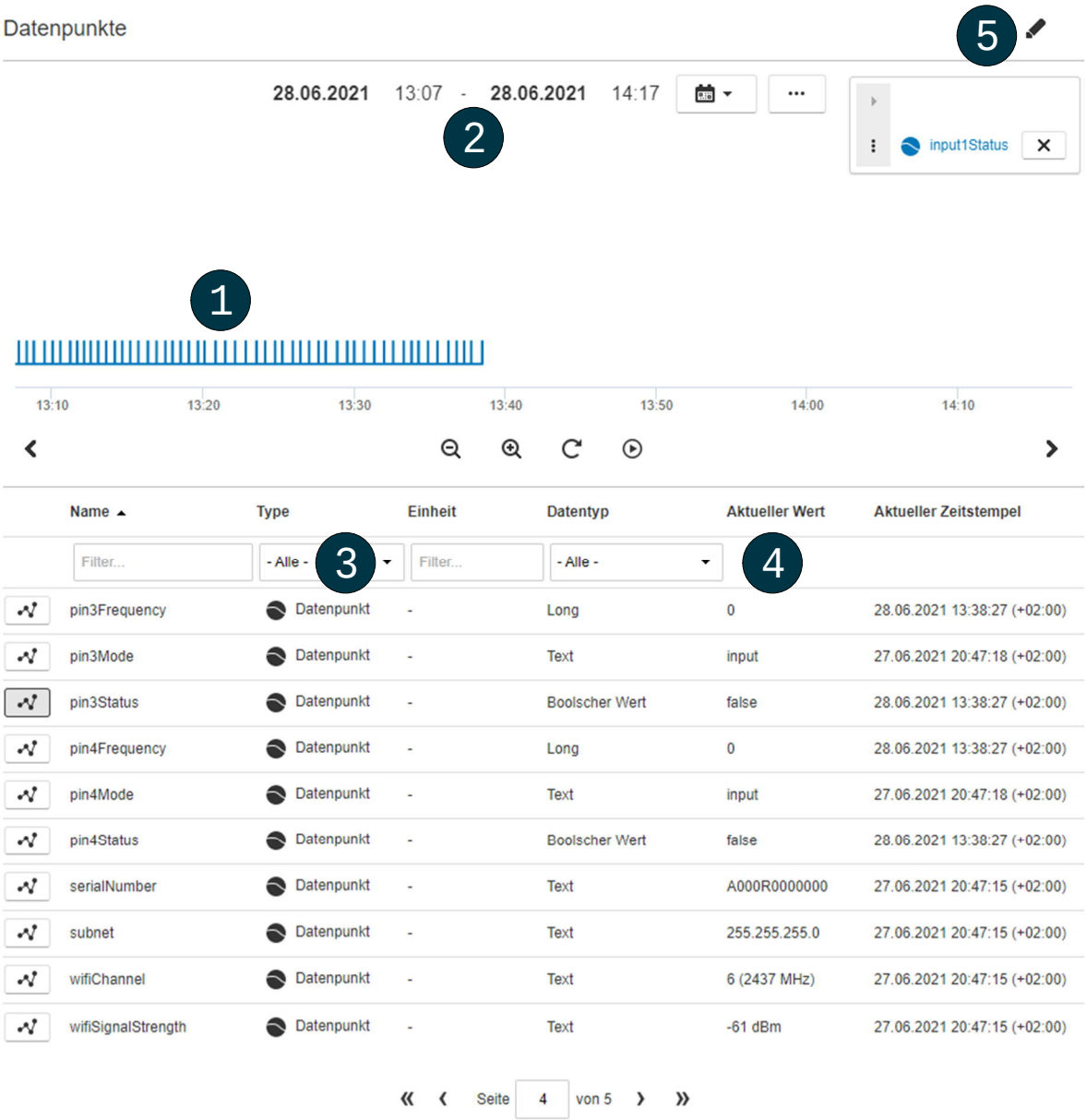
Before starting an evaluation, it makes sense to view the data stream which is supplied via the KIS.Device to the KIS.MANAGER.

In this example, a sensor is connected with Input 1 of a KIS.Device and another sensor with Input 2. If both datapoints input1Status and input2Status are selected in the datapoint overview, the time profile may look as follows:



This view of the received raw data assists in checking the calculation result plausibility later on. It is also possible to see the data range of the raw data.

Fundamentally, it is important to know that datapoints are updated on an event-driven basis. This means a KIS.Device sends an update message when something has changed.



1 Time profile of a selected datapoint

2 Setting of the display or export period → Export via ···

3 Datapoint or calculated datapoint → see Chapter 5.1

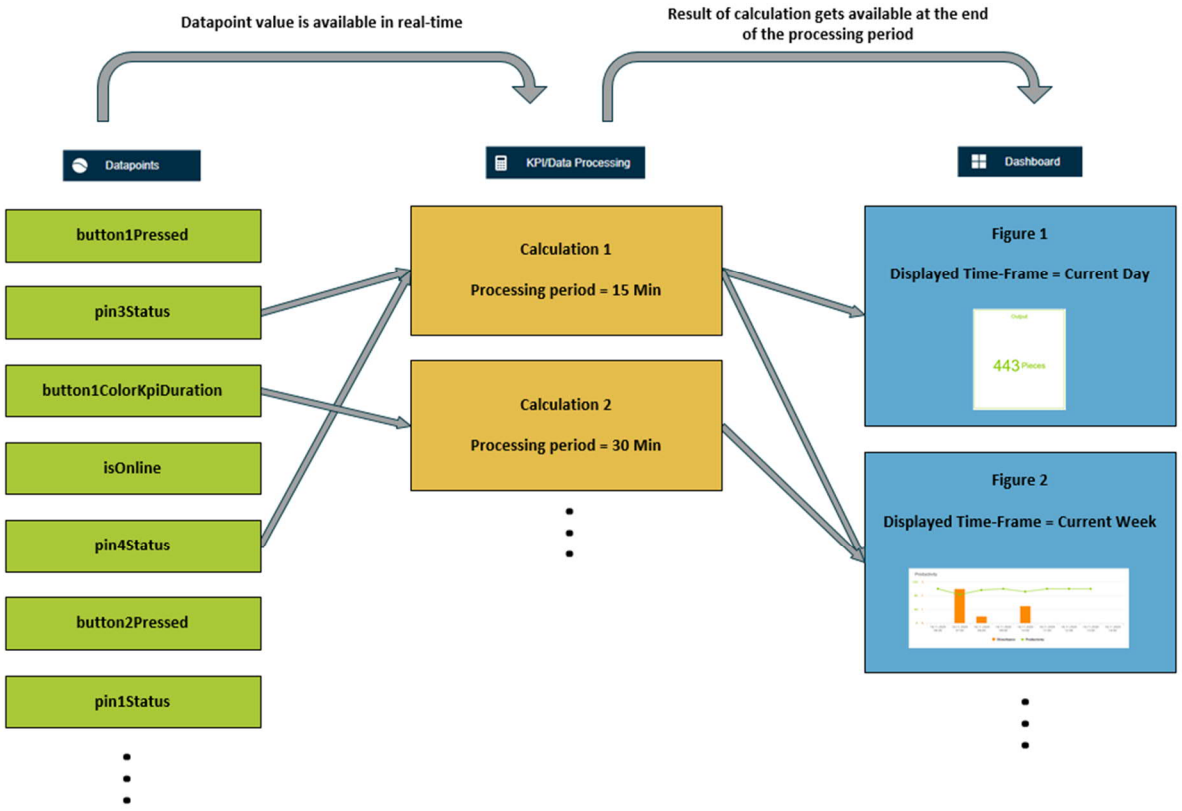
4 Current value of a datapoint

5 Delete datapoints or datapoint history

3 The route from datapoints to key figures

In order to calculate a key figure from one/multiple datapoint(s), there are three stages to run through:

- ➔ Raw values: these are extracted from the datapoints (see Chapter 2)
- ➔ Pre-processing: datapoints are converted into key figures using formulas (see Chapter 5)
- ➔ Presentation: calculation results are aggregated and displayed (see Chapter 6)

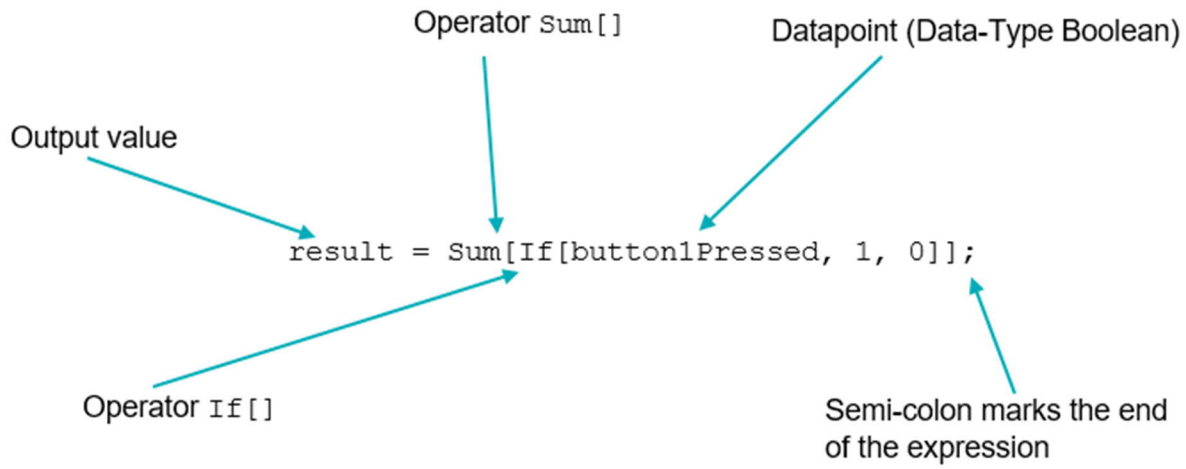


4 Fundamentals regarding the FLEX formula language

Flex provides an expression language for evaluating, aggregating and processing time-series data.

What this means in conjunction with the KIS.MANAGER:
Using the FLEX language, the continuously produced datapoint values can be compared, calculated, aggregated, and processed to produce the required key figure.

Example – Count how often button 1 has been pressed:



- The operator If[] provides a 1 each time the datapoint button1Pressed adopts the value “true” → Button pressed
- The operator If[] provides a 0 each time the datapoint button1Pressed adopts the value “false” → Button not pressed
- The operator Sum[] summates the results of the operator If[] over the set time period
- The value returned contains the results of the calculation, which can then be transferred for displaying

➔ Each row of a calculation is completed with a semicolon
➔ Spaces can be inserted anywhere without a problem
➔ The returned value can be named freely, with the exception of special characters

Fundamental options for datapoint calculation using KIS.MANAGER (detailed definition in the appendix at 8.3):

1. Numerical operations

+ , - , * , / , ^	Plus, minus, multiply, divide, power of
Round	Round to next whole number
Abs	Forms the value of the number

2. Logical operations

And	This value AND this value
Or	This value AND/OR this value
Xor	EITHER this value OR this value
Not	NOT this value

3. Data aggregation

Sum	Total
Mean	Average
Count	Condition counter
Max , Min	Maximum , minimum
First , Last	First or last value of a period
Stdev	Standard deviation
RisingEdge	Number of rising edges (digital value)
FallingEdge	Number of falling edges (digital value)

4. Comparative operations

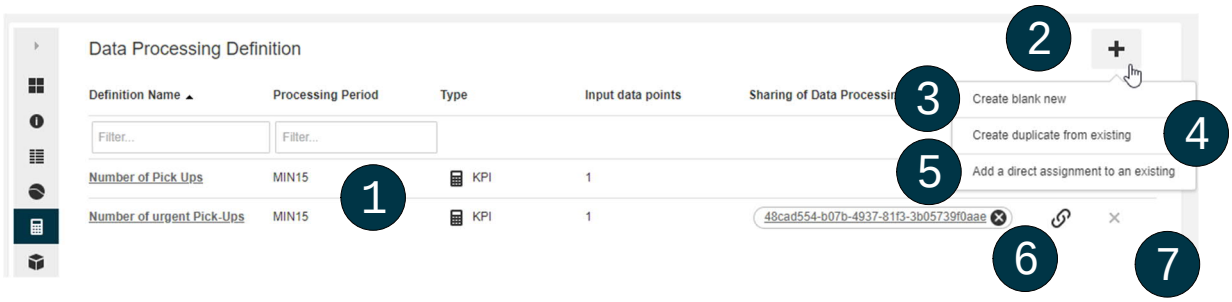
> , <	Greater than/less than
>=, <=	Greater than or equal to, less than or equal to
==	Equals
!=	Does not equal

5. Other operations

If	If, then, otherwise
Duration	Duration of a condition/state in milliseconds
Filter	Filter out implausible data
Counter	Difference between each value change

5 Calculation screen

The calculations can be created for each asset. The introduction to the calculation screens can be found at the following link:



1 List of calculations already set up

2 Clicking on + opens options to add calculations

3 Empty calculation screen opens

4 List of previously created calculations for all assets opens. If one of these calculations is selected, a (non-linked) DUPLICATE is created

➔ For details, see Chapter 7

5 List of previously created calculations for all assets opens. If one of these calculations is selected, a LINK is created

➔ For details, see Chapter 7

6 Sharing calculations → The calculation can be shared with an asset group → Each asset in the asset group is linked with the calculation

➔ For details, see Chapter 7

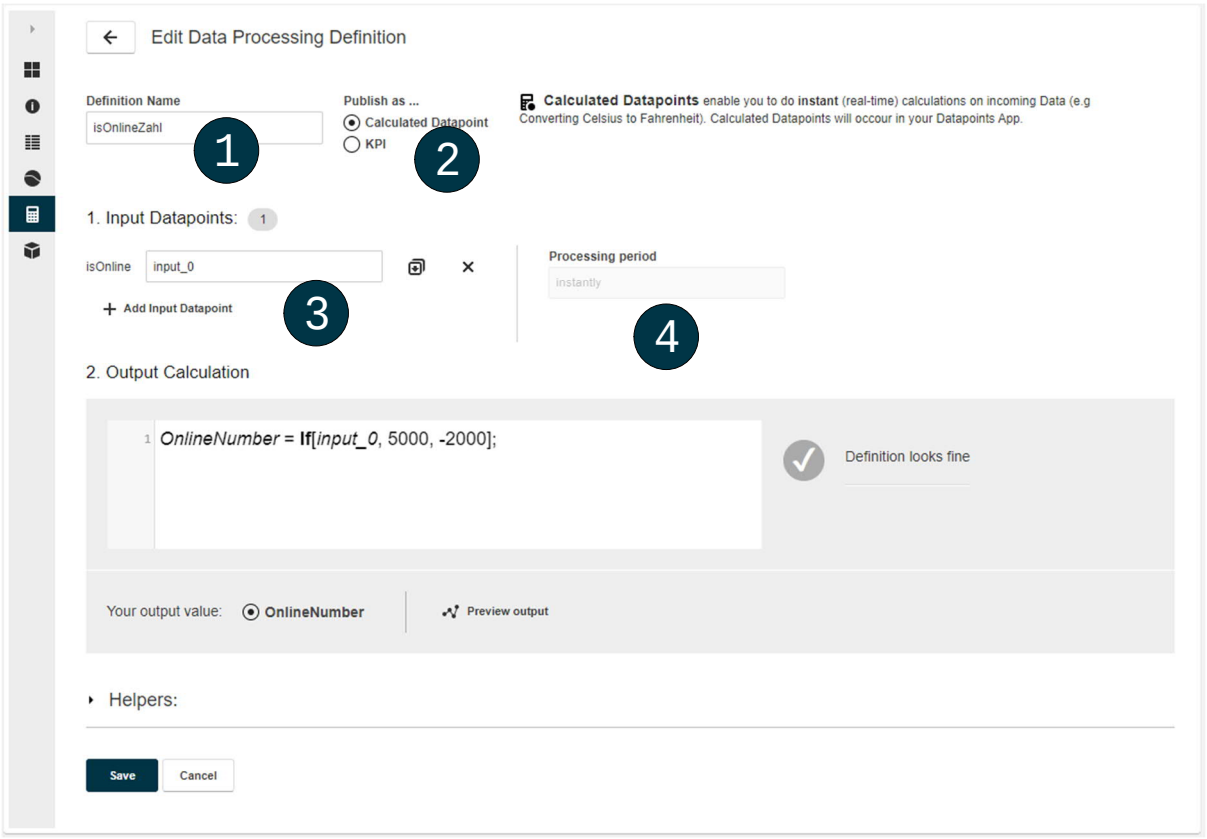
7 Deleting calculations

5.1 Setting up a calculated datapoint

The “Calculated datapoint” function is provided to allow the conversion of existing datapoints or to provide them in another form.

Example: a datapoint is available as Boolean from the device and needs to be changed into a numerical value

5.1.1 Creating an example



1 The calculated datapoint appears under this name in the list of datapoints

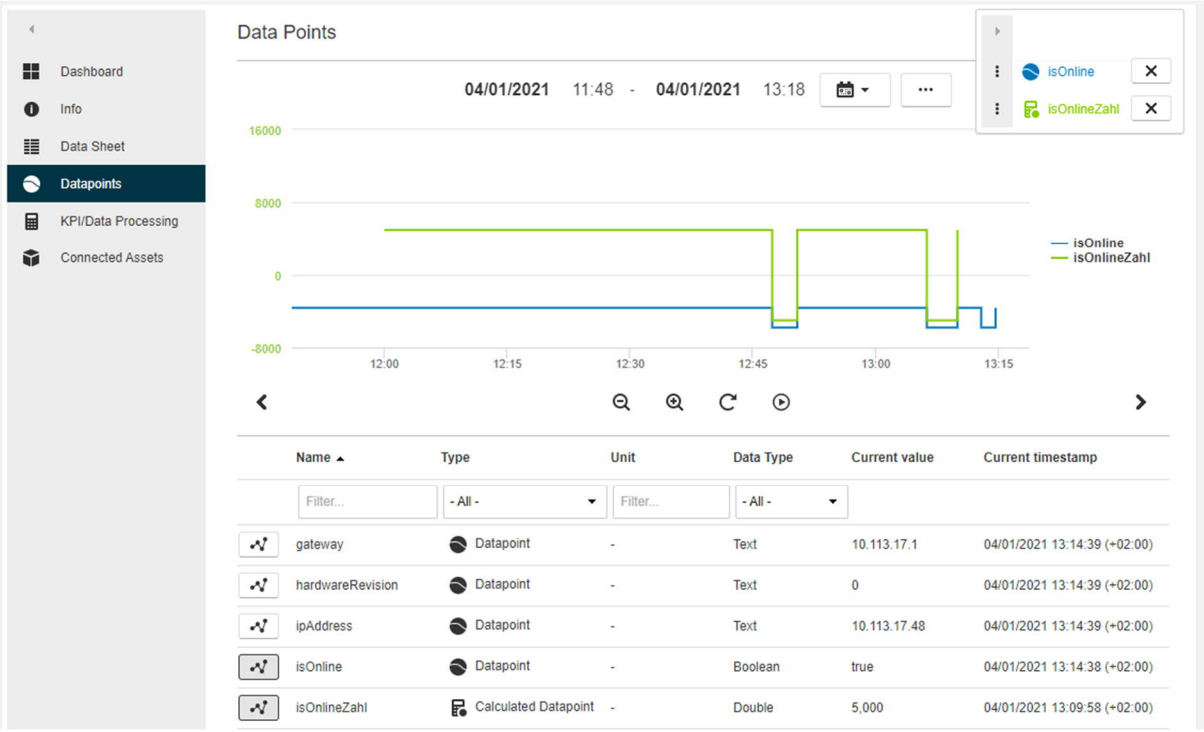
2 Selection for “Calculated datapoint”

3 Select input datapoint

4 The calculation is always executed directly when the input datapoint changes → no aggregation period can be selected

5.1.2 View in the list and option for subsequent processing

As can be seen from the datapoint overview, the newly calculated datapoint “isOnlineNumber” has appeared in the list. This datapoint can now be displayed, exported, deleted or used in KPI calculations like any other datapoint.



5.2 Setting up key figure calculations

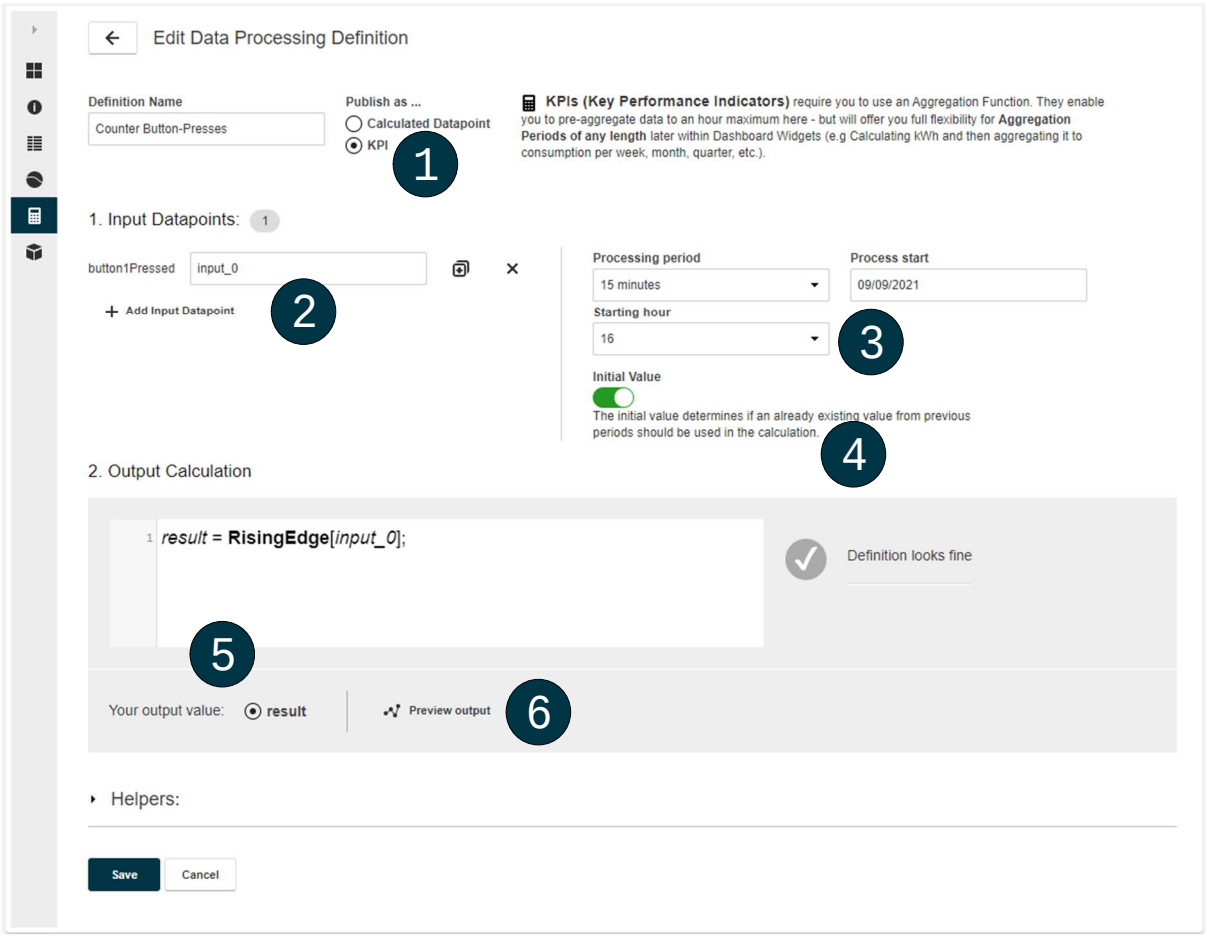
5.2.1 Differentiation to calculated datapoints

The major difference between calculated datapoints and KPIs is the time basis. Whilst calculated datapoints allow immediate conversion of a datapoint, KPIs are provided to be able to process data over a certain period.

Therefore, for KPIs so-called processing periods of 15 minutes, 30 minutes and 60 minutes are available. The processing period defines the cyclic interval at which a result is available. The result in turn is based on the datapoint values which have accumulated during the processing period.

A few example calculations can be viewed in the appendix in Chapter 8.4.

5.2.2 Creating an example



1. Selection for KPI

2. Select input data point(s) which are required for the calculation

3. Define processing period and set the desired start time of the calculation

4. If an existing datapoint value from a previous period is to be ignored, then deactivate Initial Value

5. If the calculation covers multiple partial results → select return value explicitly

6. Using Output preview, check whether the calculation provides the desired result

5.2.3 Special features

If a calculation is saved and opened again later, then the starting point is set to the current date. If this is not wanted, it must be changed manually to reflect the correct starting point. If the screen was only opened for viewing and was closed without saving, then this adjustment is not required. The date which was set the first time the calculation was saved still applies

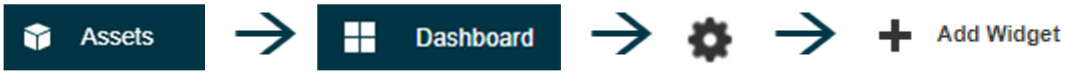
Although datapoints of the “Text” data type can be selected as an input datapoint, the formula language is not able to process their content. The datapoints with the data types “Boolean”, “Long” and “Double” can be processed

If the name of a calculation is modified and saved retrospectively, then the widgets which use the calculation must be reconfigured. Otherwise the following error is shown: “The KPI(s) with the name ´xxx` can not be loaded – please assure your configuration”

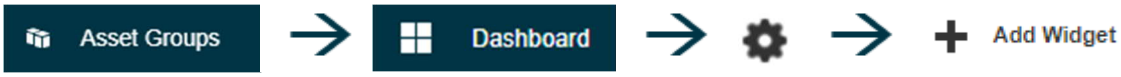
6 Presentation using dashboard widgets

Now that the KPIs have been calculated, they still need to be presented on the dashboards. The calculations are created separately for each asset. They are always available on their asset dashboards for presentation, but also on the dashboard of their asset group if the asset belongs to an asset group.

Navigation:

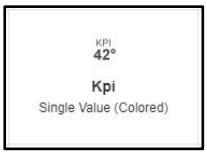


or:

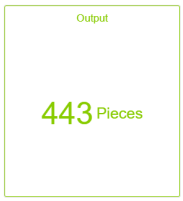


6.1 Types of diagrams

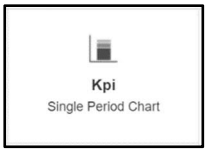
6.1.1 Single value



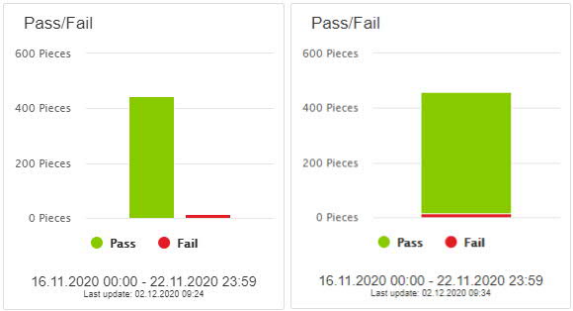
Presentation of a simple numerical value:



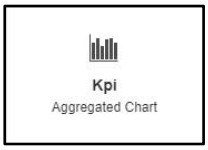
6.1.2 Single-period chart



Presentation of a stacked bar chart:

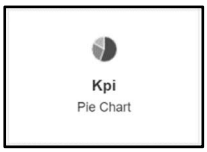


6.1.3 Aggregated chart

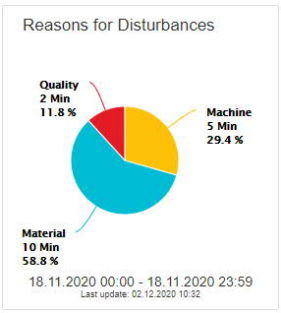


Combination of bar and line charts:

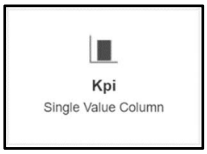
6.1.4 Pie chart



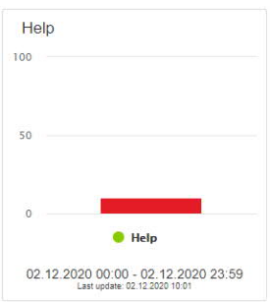
Presentation of a pie chart:



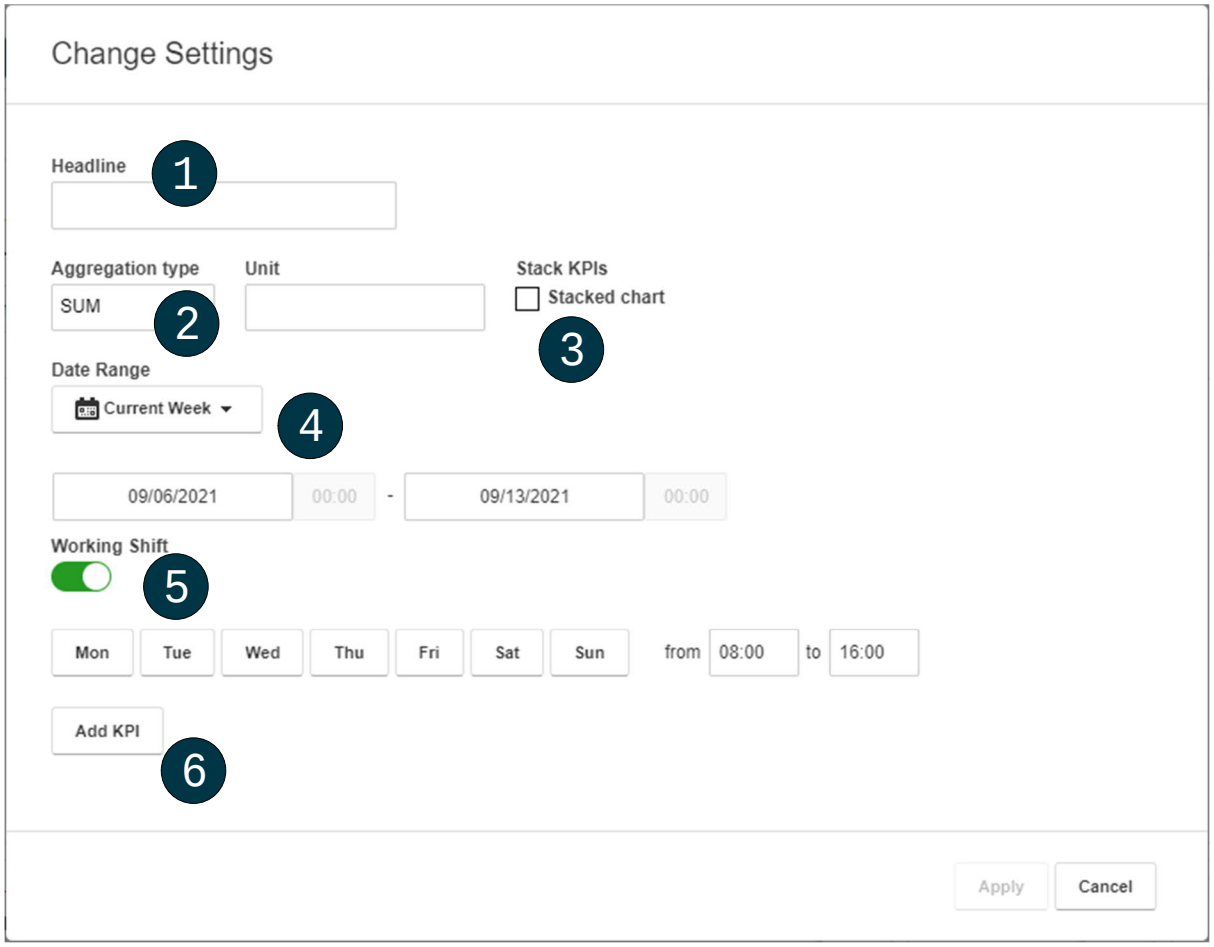
6.1.5 Single-value column



Presentation of a bar chart (threshold values for color changes are adjustable):



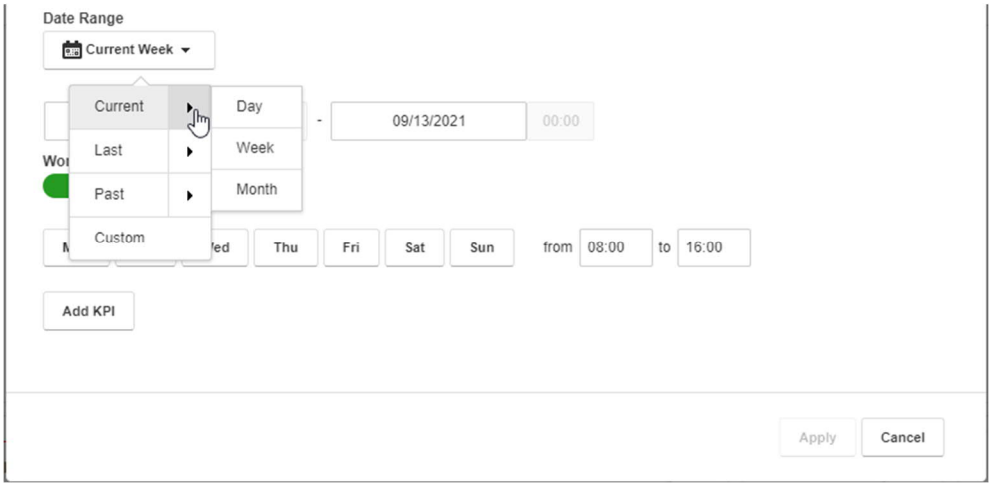
6.2 Configuration screen



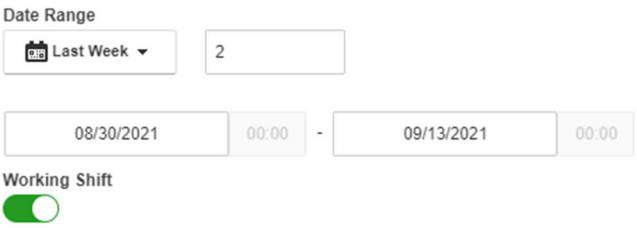
1. Title freely selectable
2. Define aggregation type (for details see Chapter 6.2.2)
3. Show diagrams stacked or next to each other
4. Select presentation time period (for details see Chapter 6.2.1)
5. Refine presentation time period based on working shifts
6. Selection of the KPI calculation(s) to be presented via this widget

6.2.1 Presentation time period

The selected period of time determines whether the available calculation results flow into the presentation.

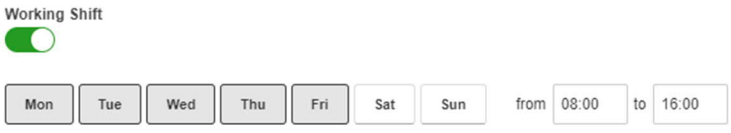


As shown in the graphic, you can choose between certain predefined time periods. This could be for example today, the last week, or the last month. If you select a period of time in the category “Last” or “Previous”, then it is also possible to decide how many past hours/days/weeks/months are taken into account:



The period of time selected is shown by the date information under the selection field. The period of time changes dynamically as time progresses (except for the “user-defined” time window).

The “Working Shift” slider allows working days and working times to be taken into account when specifying the period of time for the presentation.



In the example, only calculation results from Monday to Friday between 8 am and 4 pm in the current week are taken into account in the presentation.

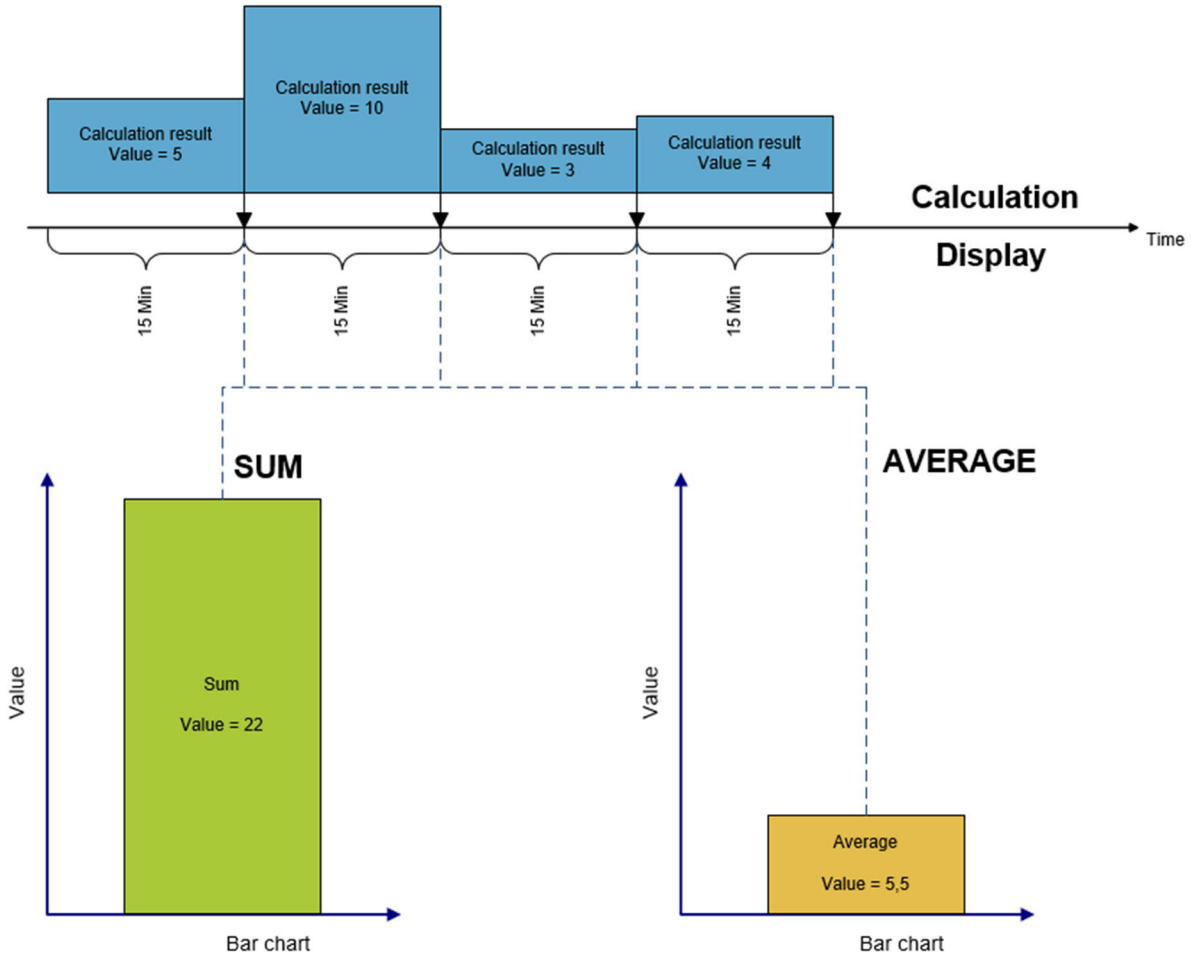
6.2.2 Aggregation

As described in Chapter 3, when calculating the KPIs, a pre-aggregation takes place over a time period of 15, 30, or 60 minutes. This is already defined when creating the KPI definition (calculation). On the other hand, for presentation, the specific time period was selected which is of interest for the user. Now the user must still decide how the calculation results are to be presented in this time period.

The following aggregation types are available:

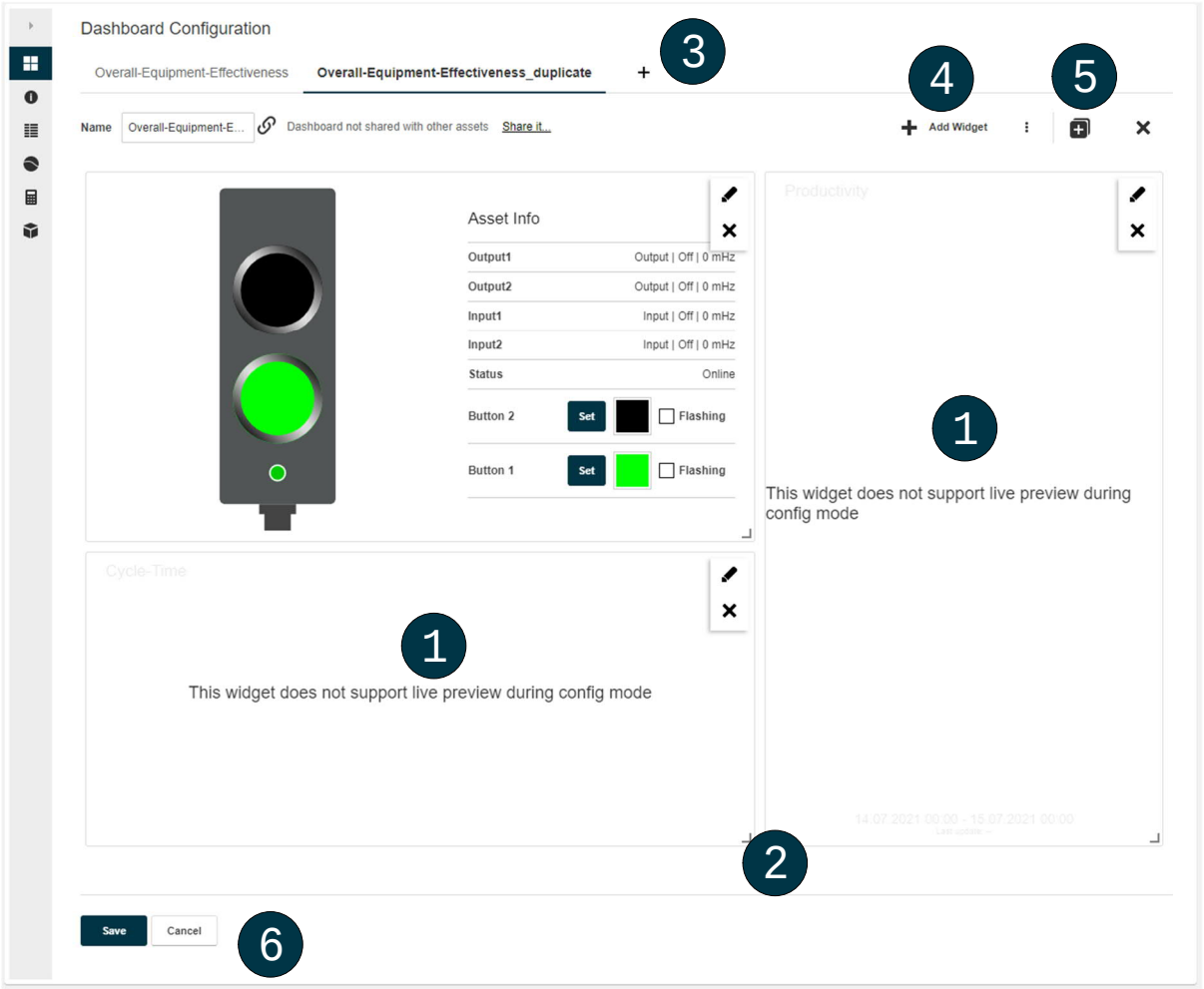
- ➔ SUM - Sum
- ➔ AVERAGE - Average
- ➔ MAX - Maximum
- ➔ MIN - Minimum

Example – presentation time period 1 hour with a calculation period of 15 minutes:



6.2.3 Setting up a dashboard

Once the KPI widget is configured, it is placed on the dashboard. Here there are still a few options for adjusting the presentation:

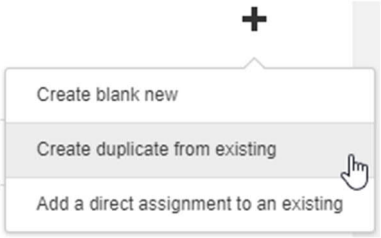


1. KPI widgets
2. Change the size of the widget by clicking and dragging
3. Add additional dashboard
4. Add additional widget
5. Duplicate dashboard
6. Save changes to dashboard

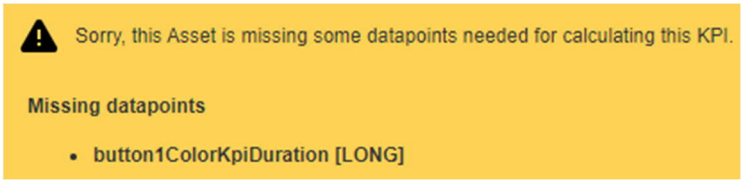
7 Practical tips Key figures

➔ **Duplicating calculations**

If a calculation is to be used multiple times, then it does not need to be created from scratch every time. Existing calculations can be duplicated – including from one asset to another.



After selection, the duplicated calculation is marked with the designation _duplicate.
If datapoints were used in the calculation which the target asset does not have, then the KIS.MANAGER highlights this with a warning:



➔ **Sharing calculations**

When calculations are duplicated, no dependency is created between the source and the target. However, when a calculation is shared, the calculation is shared and linked with a complete asset group:

The consequence of sharing is that the calculation appears in the calculation overview of each asset belonging to the asset group. It can also be opened there for editing. Since a link exists, a change has a global effect.
Even though the calculations are linked, this only relates to the formulas. The datapoints continue to be individual for each asset and a separate calculation result will be available for each asset as a result.

Sharing is ended again in the calculation overview:

➔ **Deleting datapoints**

Datapoints can be deleted partially or fully. Partially means that the history in a selected time period is deleted.

Example scenarios of when deletion would be used:

- During commissioning, implausible datapoint values occurred, which should not be included in the analysis
- A calculated datapoint is no longer required and must be removed again from the list of datapoints

If a datapoint is deleted accidentally and it is important for device functions, then this does not affect the communication between the KIS.MANAGER and KIS.Device. The history of the datapoint is lost, however during the next message between the KIS.Device and KIS.MANAGER, the affected datapoint is added once more.

8 Annex

8.1 Table with typical evaluations of datapoints

The following table lists which datapoint is typically used for which evaluation:

Datapoint	Evaluation	Data type	Special feature
button1Pressed button2Pressed	Number of button presses	Boolean	- Updates every 15 minutes
input1Status input2Status	Number of states at input Duration of states at input	Boolean	
output1Status output2Status	Number of states at output Duration of states at output	Boolean	
button1ColorKpi button2ColorKpi led1ColorKpi	Number of color states	Long	- Each color has a defined number (see 8.2) - Updates every 15 minutes (value: 99)*
button1ColorKpiDuration button2ColorKpiDuration led1ColorKpiDuration	Duration of color states	Long	- Each color has a defined number (see 8.2) - Updates every 15 minutes
isOnline	Duration of online states	Boolean	

*The datapoint values are updated every 15 minutes, even if no change has taken place. When counting color states, this can lead to incorrect results. For example, if the color state 0 (blue) is present for an extended period, then during each datapoint update it would be counted again. To avoid this, during the cyclic update the value 99 is entered instead of that of the color state. 99 means that the color state persists, however it does not trigger the KPI calculation again.

Important: this only applies for counting the color states and the datapoints listed in the table in the corresponding row.

8.2 Table for translating LED colors into numbers

To enable the use of colors in analyses, they have been assigned a numerical value. The table below shows which color belongs to which number:

Color	**RGB-hex code**	**Number**
	Example: values of the datapoint button1Color	Example: values of the datapoints button1ColorKpi and button1ColorKpiDuration
Blue	#0000FF	0
Turquoise	#00FFFF	1
Black	#000000	2
Green	#00FF00	3
Magenta	#FF00FF	4
Red	#FF0000	5
White	#FFFFFF	6
Yellow	#FFFF00	7

8.3 Operators in the FLEX language

1. Numerical operations

Function	Description	Example	Data type
`Plus[] or +`	Plus	`result = Plus[Var1, Var2]` `result = Var1 + Var2`	Double Long
`Time[] or *`	Times	`result = Time[Var1, Var2]` `result = Var1 * Var2`	Double Long
`Power[] or ^`	Power	`result = Power[Var1, 2]` `result = Var1 ^ 2`	Double Long
`Round[]`	Round to next whole number	`result = Round[Var1]`	Double Long
`Abs[]`	Value	`result = Abs[Var1]`	Double Long

2. Logical operations

Function	Description	Example	Data type
`And[] or &&`	And	`result = And[Var1, Var2]` `result = Var1 && Var2`	Boolean
`Or[] or ||`	And/or	`result = Or[Var1, Var2]` `result = Var1 || Var2`	Boolean
`Not[] or !`	Not	`result = Not[Var1]` `result = !Var1`	Boolean
`Xor[]`	Either or	`result = Xor[Var1, Var2]`	Boolean

3. Comparative operations

Function	Description	Example	Data type
`Greater[]or >`	Greater than	`res = Greater[Var1, Var2]` `res = Var1 > Var2`	Double Long
`Less[] or <`	Less than	`res = Less[Var1, Var2]` `res = Var1 < Var2`	Double Long
`Equal[] or ==`	Equal to	`res = Equal[Var1, Var2]` `res = Var1 == Var2`	Double Long
`UnEqual[]or !`	Not equal to	`res = UnEqual[Var1, Var2]`	Double Long
`GreaterEqual[]` or `>=`	Greater than or equal to	`res = GreaterEqual[Var1, Var2]` `res = Var1 >= Var2`	Double Long
`LessEqual[]` or `<=`	Less than or equal to	`res = LessEqual[Var1, Var2]` `result = Var1 <= Var2`	Double Long

4. Data aggregation (over defined “processing period”)

Function	Description	Example	Data type
`Sum[]`	Sum of numerical values	`result = Sum[Var1]`	Double Long
`Mean[]`	Mean value of numerical values	`result = Mean[Var1]`	Double Long
`Count[]`	Number of data inputs	`result = Count[Var1]`	Boolean
`Max[]or Min[]`	Maximum / minimum of numerical values	`result = Max[Var1]` `result = Min[Var1]`	Double Long
`First[] or Last[]`	First / last value of a period	`result = First[Var1]` `result = Last[Var1]`	Double Long
`Stdev[]`	Standard deviation	`result = Stdev[Var1]`	Double Long
`Percentil[]`	Calculates the defined percentile	`result = Percentil[Var1, 99]`	Double Long
`RisingEdge[]`	Number of rising edges	`result = RisingEdge[Var1]`	Boolean
`FallingEdge[]`	Number of falling edges	`result = FallingEdge[Var1]`	Boolean

5. Intervals

Function	Description	Example	Data type
`Start[]`	Returns the start timestamp of a state	`result = Start[Var1]`	Double Boolean Long
`End[]`	Returns the end timestamp of a state	`result = End[Var1]`	Double Boolean Long
`Duration[]`	Returns the duration of a condition/state in milliseconds	`result = Duration[Var1]`	Double Boolean Long
`Interval[]`	Returns the duration of set processing period in milliseconds	`result = Interval[]`	-

6. Other functions

Function	Description	Example	Data type
`Counter[]`	Returns the difference between each value change	`result = Counter[Var1]`	Double Long
`Filter[]`	Filters out implausible data	`result = Filter[Var1 > 10 && Var1 < 100]`	Double Long
`If[]`	If, then, else – based on true value	`result = If[Var1, 1, 0]`	Boolean

8.4 Example KPI formulas

Designation	Datapoints	Formula	Description
DurationColor_KIS.BOX	button1ColorKpiDuration (as “led_button1”)	`Button1Blue = Round[Sum[If[led_button1 == 0, Duration[led_button1], 0]] / 60000];`	Duration of color state in minutes (blue in this example)
DurationColor_KIS.LIGHT	led1ColorKpiDuration (as “led1”)	`LedBlue = Round[Sum[If[led1 == 0, Duration[led1], 0]] / 60000];`	Duration of color state in minutes (blue in this example)
DurationOnline	isOnline (as “input_0”)	`DurationOnline = Round[Sum[If[input_0, Duration[input_0], 0]] / 60000];`	Duration of online state in minutes
DurationOffline	isOnline (as “input_0”)	`DurationOffline = Round[Sum[If[Not[input_0], Duration[input_0], 0]] / 60000];`	Duration of offline state in minutes
DurationOutput	output1Status (as “output1”)	`DurationOutput1 = Round[Sum[If[output1, Duration[output1], 0]] / 60000];`	Duration of high state of a digital output
DurationInput	input1Status (as “input1”)	`DurationInput1 = Round[Sum[If[input1, Duration[input1], 0]] / 60000];`	Duration of high state of a digital input
CountButtonColor_KIS.BOX	button1ColorKpi (as “led_button1”)	`Button1Blue = Sum[If[led_button1 == 0, 1, 0]];`	Number of color states (blue in this example)
CountLedColor_KIS.LIGHT	led1ColorKpi (as “led1”)	`LedBlue = Sum[If[led1 == 0, 1, 0]];`	Number of color states (blue in this example)
CountButtonPresses	button1Pressed (as “button1”)	`Button1Pressed = Sum[If[button1, 1, 0]];`	Number of button presses

Designation	Datapoints	Formula	Description
CountOutputHigh	output1Status (as “output1”)	`Output1High = RisingEdge[output1];`	Number of high states of a digital output
CountOutputLow	output1Status (as “output1”)	`Output1Low = FallingEdge[output1];`	Number of off states of a digital output
CountInputHigh	input1Status (as “input1”)	`Input1High = RisingEdge[input1];`	Number of high states of a digital input
CountInputLow	input1Status (as “input1”)	`Input1Low = FallingEdge[input1];`	Number of off states of a digital input

Example_FirstPassYield	input1Status
	(as “passed”)
	
	input2Status
	(as “failed”)
		`NumberPassed = RisingEdge[passed];`
		`NumberFailed = RisingEdge[failed];`
		`Output = NumberPassed + NumberFailed;`
		`FirstPassYield = Round[100 / Output * NumberPassed];`
		`FirstPassYieldCorrected = Filter[FirstPassYield > 0];`
			Pin 3 counts good parts
			
			Pin 4 counts bad parts
			
			First-pass yield

Example_AverageCycleTime	input1Status
	(as “input1”)
		`UpTimeInput1 = If[input1, Duration[input1], 0] / 1000;`
		`DownTimeInput1 = If[Not[input1], Duration[input1], 0] / 1000;`
		`CycleInput1 = Round[Mean[Filter[UpTimeInput1 > 0]] +`
		`Mean[Filter[DownTimeInput1 > 0]]];`
			Average cycle time between two rising edges

Example_Productivity	input1Status (as “input1”)	`LineCycle = RisingEdge[input1];` `Productivity = Round[LineCycle / (Interval[] / 60000)];`	Number of parts per time unit
Example_Output	input1Status (as “input1”)	`LineCycleInput1 = RisingEdge[input1];`	Yield in pieces

9 Disclaimer

Publisher
RAFI GmbH & Co. KG
Ravensburger Str. 128-134, D-88276 Berg / Ravensburg
Tel.: +49 751 89-0, Fax: +49 751 89-1300
www.rafi.de, info@rafi.de

Exclusion of liability
All necessary steps have been taken to ensure that this document is complete and correct. If questions arise regarding specific points despite this, please contact RAFI GmbH & Co. KG or an authorized representative.

The information in this document may be changed without prior notification and does not represent any binding obligation on the part of RAFI GmbH & Co. KG. RAFI GmbH & Co. KG accepts no liability for any errors in this document.