Technical Document

Niagara KitControl Guide

March 17, 2023

Niagara KitControl Guide

Tridium, Inc.
3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

Trademark notice

Copyright and patent notice

Contents

About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document describes how to create and use the kitControl components. Sections in this guide include chapters about common kitControl tasks, concepts, and reference information. Also included are images and descriptions of the primary software user interface windows involved when working with kitControl components.

Document change log

Changes to this document are listed in this topic.

March 17, 2023

Edited the “kitControl-Ramp” component to enhance a description for “Update Interval” property.

March 24, 2020

Edited the kitControl-BqlExprComponent to add a description for Expr property, plus added data for bql expr and like query with wild card details.

June 16, 2019

Updated the “kitControl-And” topic to clarify handling of "null" value.

June 07, 2019

Initial release.

Related documentation

Additional information is available in the following documents.

- *Getting Started with Niagara*
- *Niagara Platform Guide*

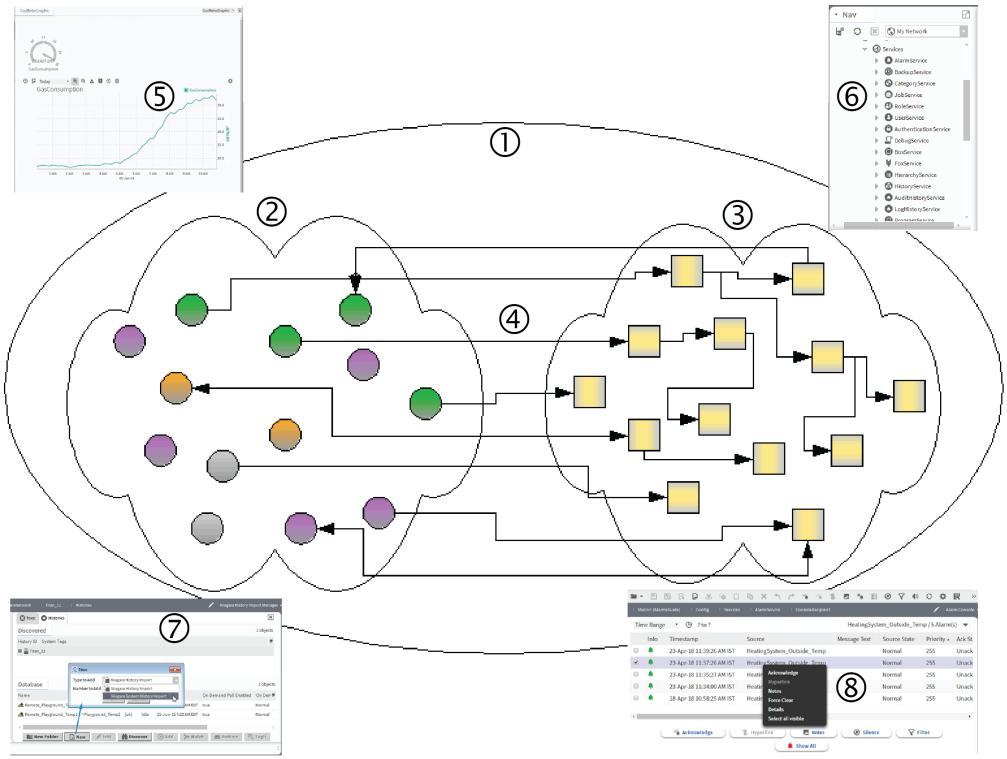
Chapter 1 About kitControl

Topics covered in this chapter

- About the kitControl palette
- Where to locate kitControl components
- Extensions and kitControl components

The kitControl palette contains various components that you can use in combination with points, both simple control points and proxy points. Whereas proxy points read (and possibly write) data from (and to) remote devices, kitControl components provide the data manipulation building blocks that let you further process point data. Included are HVAC components like a PID loop and sequencers, a variety of Boolean logic components, math components for numeric values, and miscellaneous components. Together with proxy points and schedules, kitControl components provide the basic set of the common components for modeling control logic.

Figure 1 Conceptual use for kitControl components in a building model



The usage of kitControl components is entirely optional. It is possible to build a monitoring-type application with only proxy points (and perhaps a few simple control points). This would allow real-time data monitoring, plus user-invoked action overrides through writable points' right-click command menus. As needed, you could also add extensions to the proxy points for alarming and history collections.

(1) represents the common component model. Inside the model are three things:

- (2) station points and proxy points
- (3) kitControl components.
- (4) Links between the points and kitControl components.

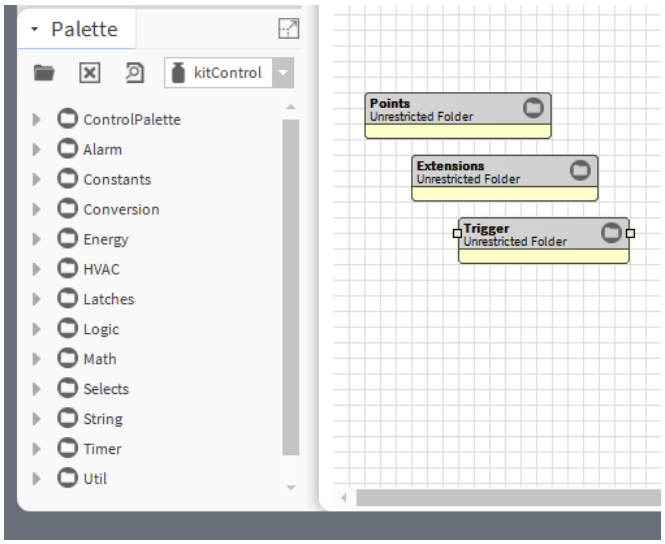
Outside the common component model are:

- (5) Px graphics
- (6) Services
- (7) Histories
- (8) Alarms

About the kitControl palette

The kitControl palette provides a toolkit with which to manage building automation systems. The palette is a standard feature of .

Figure 2 Folders in the palette kitControl



The kitControl palette contains over 90 unique components across 12 folder categories (not counting ControlPalette). The folders in the palette reflect the types of components:

- ControlPalette includes these subfolders: **Points**, **Extensions**, and **Trigger**. These standard control palette folders in the kitControl provide easy access to the simple control points, extensions, and timers, in addition to the kitControl components found under its other folders.
- Alarm contains three extensions for alarming. One is expressly for a LoopPoint (setpoint-deviation) alarming. The others provide alarming options for a Boolean point with a DiscreteTotalizerExt. A fourth component provides alarm count monitoring of any Alarm Class, and includes a Boolean relay output.
- Constants contains four components, one for each data category. Each provides a linkable status-type output, and a Set action for changing value.
- Conversion contains 19 components that mainly convert status values to simple values, and vice versa. This folder also contains other special conversion types.
- Energy contains 10 components for typical energy functions, such as degree day calculation and electrical demand limiting.
- HVAC contains nine components for typical HVAC functions, such as interstart delay, lead-lag control, and sequence control. This folder includes Tstat (thermostat) and LoopPoint (PID loop) components.
- Latches contains four components, one for each data category.
- Logic contains 10 components, each with StatusBoolean output.
- Math contains 23 components for processing one or more numeric input values to produce a StatusNumeric output.

- Selects contains four components, one for each data category.
- String contains six components with one or more StatusString inputs.
- Timer contains five components. Three are timers: BooleanDelay, NumericDelay, and OneShot. Two are absolute timers: Current Time, and TimeDifference.
- Util contains 16 various utility components including Expr (BQL Expression).

Where to locate kitControl components

As with simple control points, you can copy kitControl components to any folder or component in the station. This includes placing kitControl components under any device Points extension (container), or under any Points extension subfolder.

There are two competing best practice philosophies about locating your control logic:

- Philosophy A maintains that only proxy points belong under a device's Points container. You can add extensions (control, history, and alarm), as needed, to proxy points. However, other components (kitControl components, simple control points, schedules) should be located under a central folder—not under the station's Driver architecture—but instead in subfolders under a main **Logic** folder (by convention) created in the root of the station's **Config** container.

  This philosophy requires many internal links between proxy points and kitControl components. It often makes following control logic harder, because you do not see most links except as knobs. In addition, it makes applications much less portable in that you cannot copy an entire application by selecting a single device container.
- Philosophy B maintains that kitControl components and schedules belong under each device's **Points** container (either directly, or in subfolders). This allows you to create local links between device proxy points and other components. Copying an entire application requires copying a single device container.

The original intention of Philosophy A was to establish a logic convention to simplify universal support of different application types. However, Philosophy B offers easier portability of each application, allowing easy replication and reuse at a device level. In general, use of Philosophy B is more common. In other words, it is preferable to locate kitControl components where they are needed.

Extensions and kitControl components

You can add point extensions, such as alarm extension, history extension, or perhaps a control extension, to many kitControl components. Other kitControl components cannot receive extensions.

These are some specific examples of kitControl components that support extensions:

- Average object (**Math** folder): Inputs link to multiple proxy NumericPoints, each representing a room temperature. The Average object represents a Zone temperature (average). To this object you may add an alarm extension (OutOfRangeAlarmExt) and history extension (NumericInterval).
- And object (**Logic** folder): Inputs link to multiple proxy BooleanPoints, each representing fan status (off or on). The And object links to downstream control logic. To track when all fans are running, you may add a history extension (BooleanChangeOfValue) and perhaps a control extension to collect runtime (DiscreteTotalizerExt).
- NumericSwitch object (**Util** folder): Inputs link to two proxy NumericWritables, each representing a power rate (kW). The object output links to downstream control logic (and represents the current switched rate). To totalize this effective rate into energy accumulation, you can add a control extension (NumericTotalizerExt) and proper scaling to collect kWh.

These components are not based on simple control points (ControlPoint) and do not support the addition of extensions:

- Constants components (any)
- Conversion components (any)

- Energy components (any)
- HVAC components (except for LoopPoint, InterstartDelayControl and Tstat, which do support extensions)
- Latches components (any)
- Selects components (any)
- String components (any)
- Timer components (any)
- Util components (except BooleanSwitch, EnumSwitch, MultiVibrator, numericSwitch, Ramp, Random, and SineWave, which do support extensions)

You can quickly tell if a kitControl object can receive an extension, by seeing if it has the frozen ProxyExt (proxy extension). To see this, expand the object in the kitControl palette, Nav tree or view the object's property sheet. If a **`Proxy Ext`** property is present, you can add other extensions (provided they are the correct type), for example a BooleanChangeOfValue history extension for a Logic-type object, and so forth. If you try to add an extension to a kitControl component that does not support extensions, you receive an illegal parent error message.

NOTE: The frozen ProxyExt is the only use for the proxy extension in a kitControl object. Its value is always null. Only control points can be proxy points.

Chapter 2 Building management examples

Topics covered in this chapter

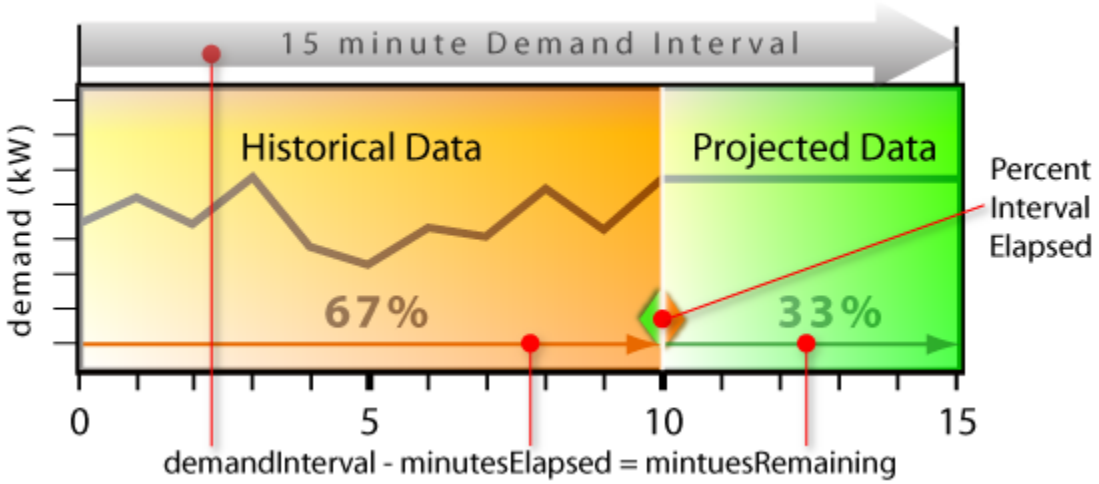
- How to project demand for electricity
- How to manage demand for electricity
- Lead-lag compensation example – kitControl
- PID loop configuration
- Night time heat configuration
- Reversible actuator control
- Heating and cooling management to reduce energy consumption
- Hot water control setpoint based on outside air temperature
- Data gathering

The kitControl components facilitate the flexible modeling of building automation systems. This chapter provides practical examples to help you creatively construct an efficient model for your application.

How to project demand for electricity

The kitControl's ElectricalDemandLimit component monitors instantaneous electrical power, calculates a projected demand average over a specified demand interval, and directs the shedding of specific loads when the projected demand average exceeds a specified demand limit. As projected demand levels recede, the component prioritizes the restoration of loads. In addition, it records and saves peak demand times, dates and values for both the current and previous months.

Figure 3 Demand interval illustration



On the left are actual historical electricity demand data. On the right are projected data. The component uses the two data sets across a 15 minute demand interval to calculate the **`Projected Demand Average`**. In this example, the **`Percent Interval Elapsed`** property is set to 67. This causes the actual minutes elapsed over the 15 minute demand interval to equal 10 minutes and the minutes remaining to equal five minutes. The component averages the actual historical demand data over 10 minutes and uses the instantaneous demand taken at minute 11 to calculate an average projected data value for the remaining third of the period. Using these two numbers, it calculates the **`Projected Demand Average`**.

You configure Percent Interval Elapsed on the property sheet of the ElectricalDemandLimit component.

Using mostly projected values

This example uses mostly projected values to calculate the **Projected Demand Average**.

Assuming that the **Demand Interval** is set to 15 minutes and the **Percent Interval Elapsed** property is set to 7 (7%), the system calculates demand based on the current minutes demand reading which it projects for the remaining minutes in the **Demand Interval** window. The following example assumes that the **Power Input** value is 400.

`minutesElapsed = 15 * 6.67 / 100 = 1`

`minutesRemaining = 15 - 1 = 14`

`calculated total = (current Power Input * minutesRemaining) + Power Input from each minutesElapsed interval`

`calculated total = (400 * 14) + 400 = 6000`

`projectedDemand = calculated total / (minutesElapsed + minutesRemaining)`

`projectedDemand = 6000 / (1 + 14) = 400`

By setting the **Percent Interval Elapsed** property to a value that corresponds to the first minute of the demand window, the system calculates demand based almost entirely on a projected data value.

In this case, the projected demand calculation uses the **Power Input** value at minute two (2) and averages demand across the remaining 14 minutes.

Using all recorded values

This example uses mostly recorded values to calculate the **Projected Demand Average**.

Assuming that the **Demand Interval** property is set to 15 minutes and the **Percent Interval Elapsed** property is set to 93%, the system calculates demand based completely on recorded demand readings for the current minute and the 14 previous minutes. Following, is an example of using all recorded values and no projected values.

```
minutesElapsed = 15 * 93.33 / 100 = 14
minutesRemaining = 15 - 14 = 1
calculated total = (current Power Input * minutesRemaining)
+ Power Input from each minutesElapsed interval
calculated total = (600 * 1) + 600 + 600 + 600 + 600 + 600
+ 600 + 600 + 600 + 600 + 400 + 400 +400 + 400 + 400 = 8000
projectedDemand = calculated total / (minutesElapsed
+ minutesRemaining)
projectedDemand = 8000 / (14 + 1) = 533
```

In this case, there would be no projected demand. Actual demand is used in this calculation.

This example assumes that the **Power Input** is currently 600 and has been at that level for the previous nine minutes, prior to that, the value was 400.

By setting the **Percent Interval Elapsed** to a value that corresponds to the last minute of the **Demand Interval**, the projected output is a sliding window average of the minutely recorded **Power Input** values.

Using recorded and projected values

This example uses projected and recorded values to calculate the **Projected Demand Average**.

The EDL (Electrical Demand Limit) component defaults to a **Percent Interval Elapsed** property value of 75%. This setting configures the component to calculate demand based 75% on actual recorded **Power Input** property values and 25% on a projection, which assumes that demand will remain at the current level for the remaining minutes in the **Demand Interval**.

How to manage demand for electricity

It is one thing to project future electricity demand, but how does the framework then manage the demand for power to reduce cost?

The framework divides a day into three periods. For each period, you configure a specific demand limit. The framework compares the **Projected Demand Average** to this **Demand Limit** at the current time of day and, based on this comparison, sheds or restores electrical load as appropriate. In other words, if the projected demand is higher than the demand limit for the current time of day, the system sheds power. If shedding is active, and the projected demand is lower than the demand limit for the current time of day, the system restores power.

Thirty-two power shed levels serve the shedding calculations. Each shed level value represents an amount by which to drop the power when that shed level is active. You configure these estimates manually in the ElectricalDemandLimit and ShedControl property sheets. They do not reflect live data. When the projected demand exceeds the **Demand Limit** for the current demand period, the component calculates how many loads to shed. You should set the **Power Shed Level** properties to demand values based on loads that are controlled by the specific shed group. The load shed logic calculates how much demand to reduce, and then uses the **Power Shed Level** values to determine how many of the loads to shed.

Shedding and restoring loads occurs in a fixed priority. The system sheds Power Shed Level1 first and restores it last.

Electrical demand reduction calculations

When **Projected Demand Average** exceeds the **Demand Limit** value for the current **Demand Interval**, the system calculates how many loads to shed.

You should set all available (or desired) **Power Shed Level** properties (1-32) to a demand value based on the loads that are controlled by the specific shed group.

An estimate of the demand associated with a group of equipment can be calculated if the operating voltage and current draw are known for the loads. This is an example of a calculation for single-phase loads:

`W = V * A`

`W = 120 Volts * 30 Amps = 3600 Watts = 3.6 kW`

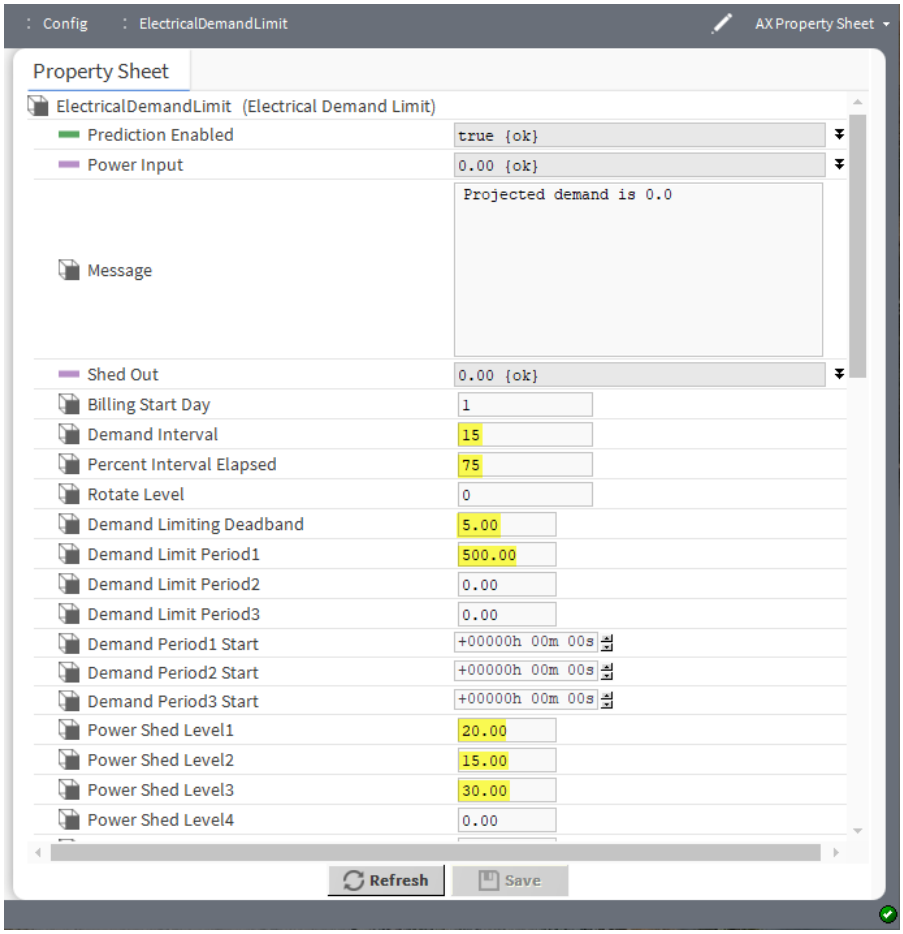
The following is an example using the square root of 3 for three-phase loads:

`W = V * A * 1.73`

`W = 480 Volts * 30 Amps * 1.73 = 24919 Watts = 24.9 kW`

Assume that the current demand limit period is Demand Limit Period1, and the ElectricalDemandLimit component is configured as follows:

Figure 4 EDL example configuration



In addition, the calculated total is 7625 kW and the Projected Demand Average is 533 kW. These equations show the kW that needs to be shed:

`targetIntervalTotal = demandLimit * (minutesElapsed + minutesRemaining)`

`targetIntervalTotal = 500 * (11.25 + 3.75) = 7500`

`powerChange = (calculated total - targetIntervalTotal) / minutesRemaining`

`powerChange = (7625 - 7500) / 3.75 = 33.33 KW that needs to be shed`

Since **`Power Shed Level1`** is only expected to reduce the demand by 20 kW, both **`Power Shed Level1`** and **`Power Shed Level2`** must be shed to reduce the demand by an expected 35 kW combined.

The necessary loads are shed in sequential order during the same execution cycle, without evaluating the actual impact on demand.

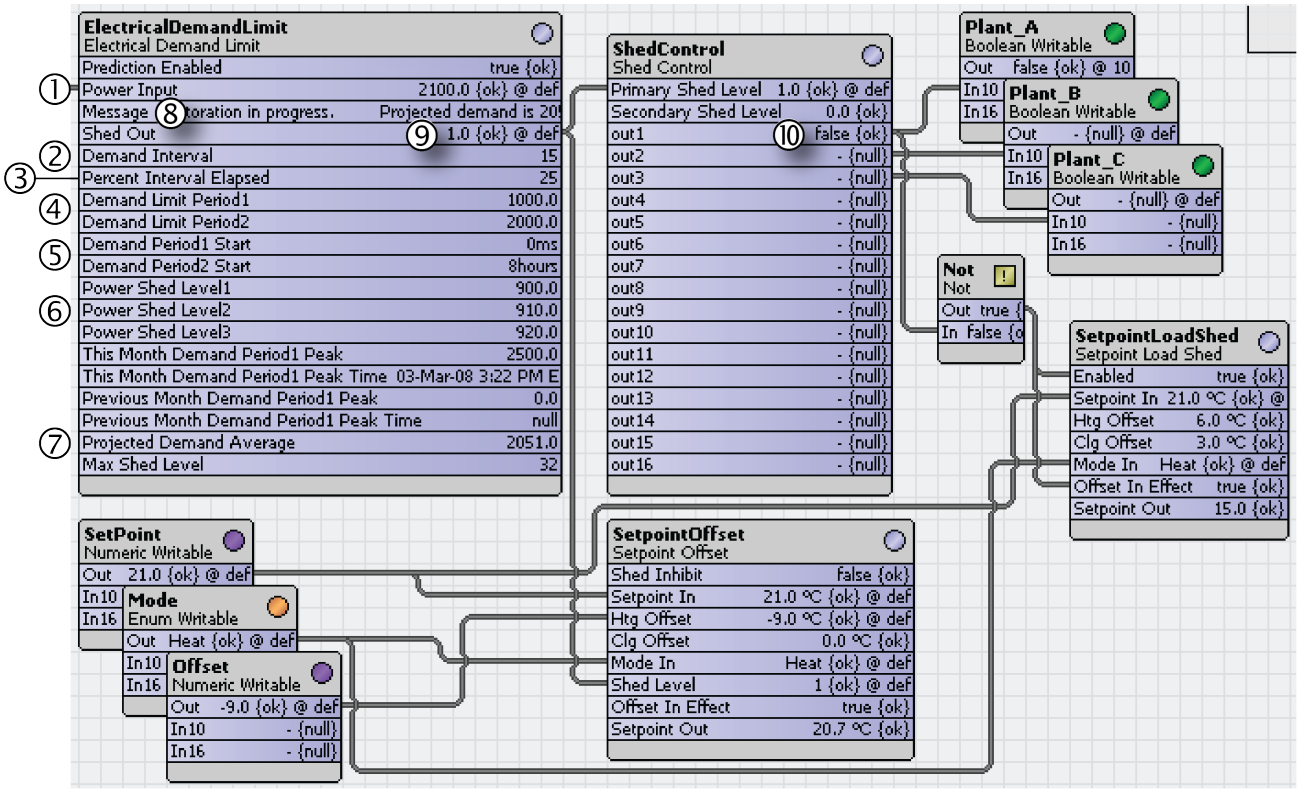
The load shed determination is based on the projected reduction in demand for each group.

Subsequent calculations may result in additional load shedding if the actual demand is not reduced below the demand limit.

Electrical demand reduction examples

This example shows a partial wiresheet view of an ElectricDemandLimit component configured for shedding energy loads. The current time period in this example is **`Demand Period2`**.

Figure 5 Wiresheet view of an EDL example application



Power Input (1) reflects total demand, which is linked into the EDL component **Power Input** property. This is a single input value, therefore the power sources need to be totaled before linking because there is more than one meter supplying actual electrical demand data. This demand level value is expected to change in response to load shedding.

Demand Interval (2) is set to update every 15 minutes.

Percent Interval Elapsed (3) is set to 25 percent, which adds more weight to current demand and less weight to historical demand for each calculation of the **Projected Demand Average.**

Demand Limit Periods 1 and 2 (4) are set to 1000 and 2000, respectively. These values specify the demand levels that initiate power shedding.

Demand Period (5) start times for periods 1 and 2 are set to 0000 (midnight) and 0800 (8 am) respectively. **Demand Period3 Start** time is not shown. These times specify the start time for each of the demand periods.

Power Shed Levels1–3 settings (6) are based on these estimates:

- By shedding loads associated with Power Shed Level 1, the amount of demand will decrease by 900 kW.
- Power Shed Levels 2 and 3 should be set at 910 and 920 respectively.

If these estimates are correct, shedding at level 3 reduces demand by the sum of all three shed levels: `(900 +910+920)=270330 kW`.

Projected Demand Average (7) has a current value of 2051.

Since this value is greater than the demand limits, the component initiates shedding and displays this message (8): "`SHEDDING REQUIRED ! Projected demand is 2051`".

The **Shed Out** (9) value of 1.0 indicates that one level is to be shed. This property links to a Shed Control component, which configures the specific Shed Level (1–16) links into Boolean controls. In the example, these controls shut off power to Plant_A, Plant_B, and Plant_C, with Shed Level(1, 2, and 3), respectively. In

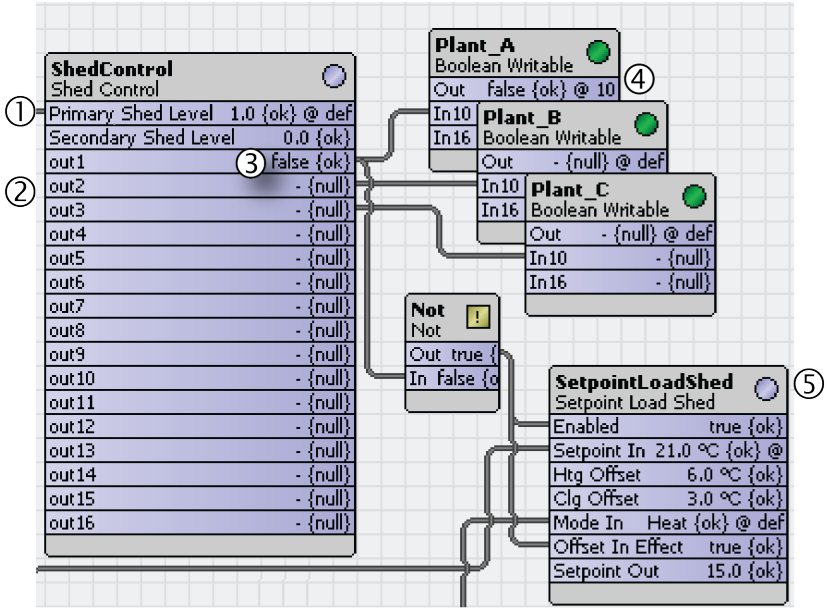
addition, the ShedControl component's **out1** value (10) links to a SetpointLoadShed component that uses a setpoint offset to reduce power usage.

You can also link from the **Shed Out** property to other energy components, such as a Setpoint Offset component (also shown here).

With a Shed Level1 in effect, Power Input is at 2100, and the Projected Average Demand value is 2051, still greater than the Demand Limit Period2 value of 2000. The Power Shed Level2 value (estimate) indicates that invoking a Power Shed Level2 yields a decrease of 910kW and brings the demand down below the limit. If this estimate is fairly accurate, actual power usage should drop and the Power Input value should lower to below the Demand Limit Period2 value.

The following wire sheet provides a close-up example of how to configure the ShedControl component.

Figure 6 ShedControl component example



The **Primary Shed Level** (1) links from an EDL component. Its value is 1.0. No Secondary Shed Level is used.

Out1, 2, and 3 (2) link to Boolean controls that are set to turn off power to Plant_A, B, and C, respectively.

The current **Primary Shed Level** of 1.0 sets out1 (3) to `false` and the Plant_A Boolean control status (4) to `false`. This should turn off power to any power consuming devices that are linked to this object.

The out1 value is also linked to the SetpointLoadShed component (5), which adjusts a setpoint to reduce the electrical demand.

If more shed levels are needed, the component sets out2 and out3 to `false`. If shed levels are no longer needed, the component invokes a restoration, which restores out3, out2, and out1, in that order, to null as allowed by the current shed level.

Reducing demand for electricity

Configuring shed and restoration control using the ElectricalDemandLimit component can reduce peak demand for electricity and save money.

Step 1 Drag the ElectricalDemandLimit component to the Nav tree or wire sheet and open its property sheet by double-clicking it.

Step 2 Enable the component by either linking to it in the wire sheet or by manually setting the **Prediction Enabled** property value to `true`.

Step 3 Configure the component properties.

Step 4 Set up any power input links including linking the Shed Out slot (output) to a ShedControl object.

This provides equipment shed and restoration control.

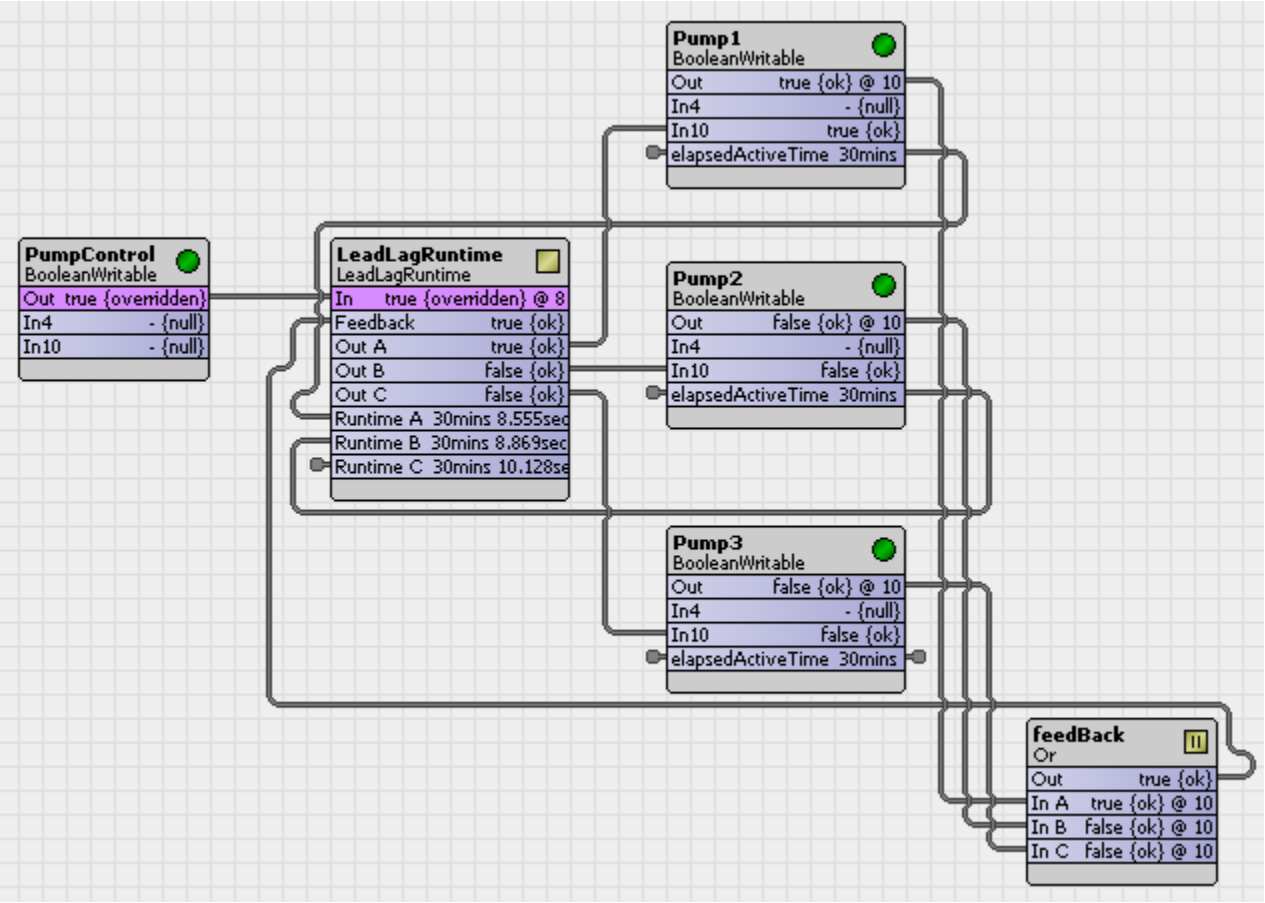
Based on how you configure the EDL component, the calculations direct that load shedding, load restoration, or no action be taken. With each calculation, the component updates and displays the projected average demand.

Lead-lag compensation example – kitControl

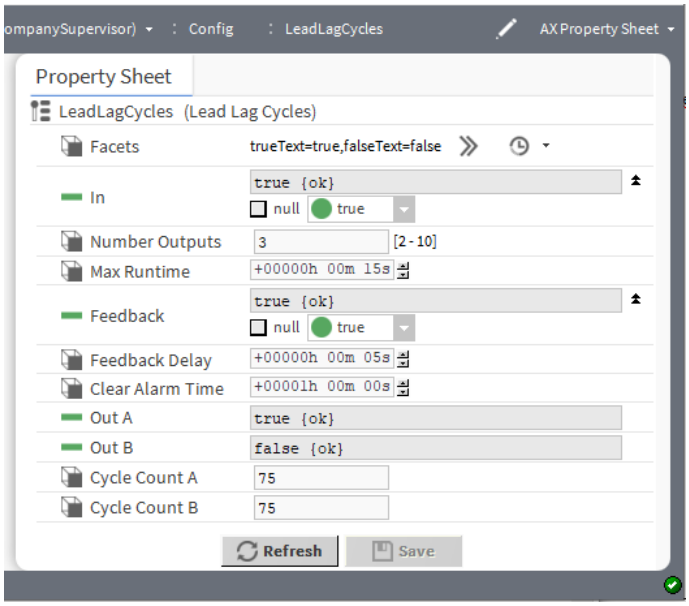
This topic provides an example of how to use the LeadLagCycle component to control when three pumps run. The purpose of controlling the three is to prevent them from drawing excess electricity by all three running at the same time.

The wire sheet for controlling the lag times for three pumps might look like this:

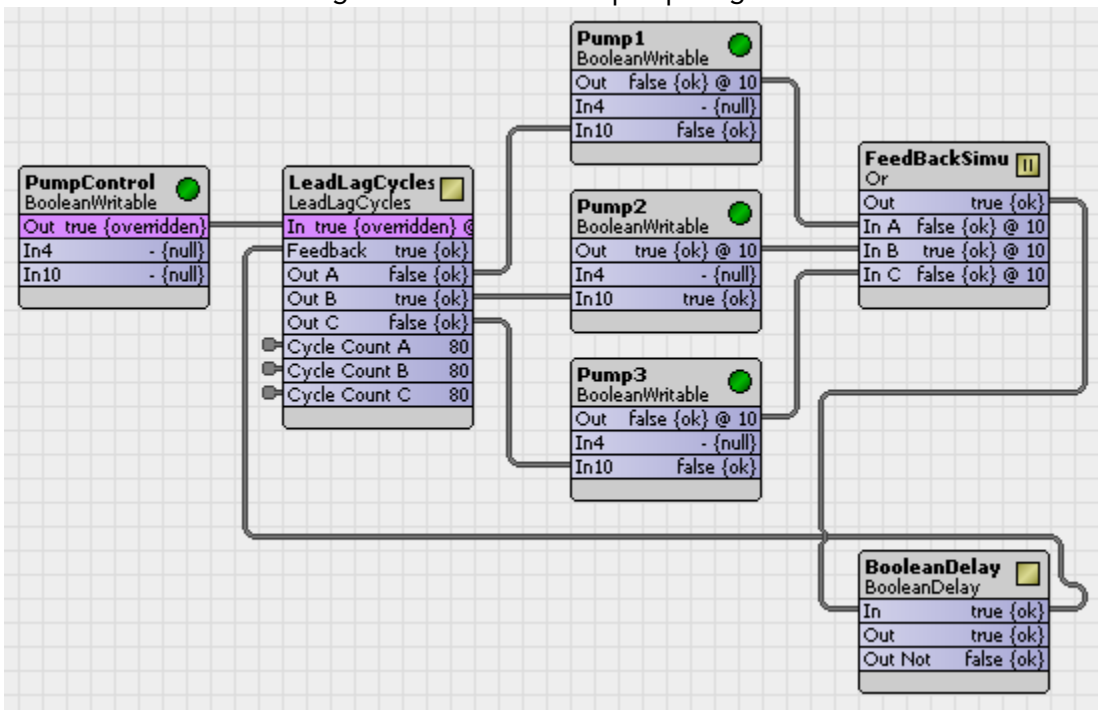
Figure 7 LeadLagCycle example with linked objects



Each of the three BooleanWritable points has a DiscreteTotalizerExt with its changeOfStateCount slot linked back to a Cycle Count x input on the LeadLagCycles object (not visible in the illustration). The feedback Or object simulates feedback, fed through a BooleanDelay object. The following shows the configuration of the property sheet:

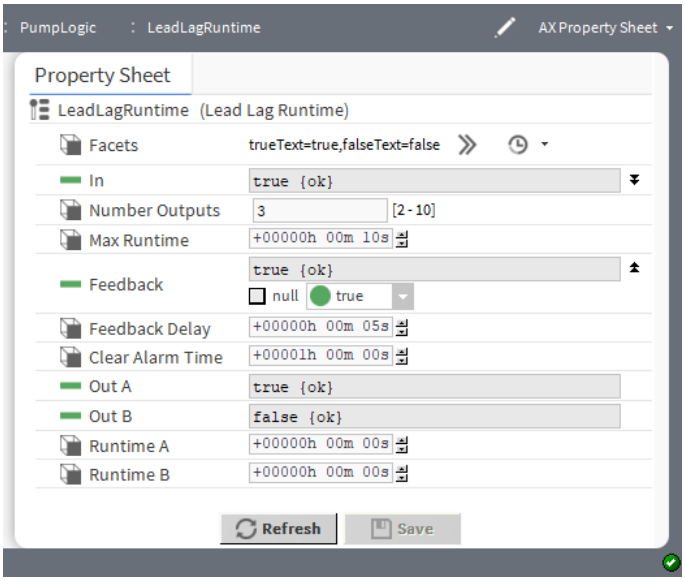


The wiresheet for controlling the runtime for three pumps might look like this:



Each of the three BooleanWritable points has a DiscreteTotalizerExt, with its elapsedActiveTime slot (1) exposed up in the composite of the parent point for link clarity. The feedback Or object (2) simulates feedback. The property sheet would look like this:

Figure 8 LeadLagRuntime properties



PID loop configuration

The LoopPoint component implements a simple PID (Proportional, Integral, Derivative) control loop.

The following terms are used when describing the operation of the LoopPoint component.

- Process variable (PV): the controlled process, in other words, the value at the setpoint input, your starting point.
- Setpoint (setpt): the target for the process variable. In other words, the value at the setpoint output, your goal.
- Setpoint error(E_S): the difference between the process variable and the setpoint acted upon by the loop algorithm.
- Loop output: the correction signal produced by the loop algorithm. This value should be linked directly or indirectly to the NumericWritable component used to position a proportionally-modulated device, such as a valve or damper, that controls the process variable.
- Proportional gain (K_P): the value of the property **`Proportional Constant`** sets the overall gain of the loop as in this ratio:

  K_P = `Output range / affected process range` (sometimes called throttling range)
- Throttling range: the amount of process variable change expected as a result of throttling the system between the **`Minimum Output`** and **`Maximum Output`**.
- Bias: the algorithm typically adds this value to the output to correct any offset error in proportional calculations and only as a control pivot output value for when PV = setpt.
- Action: defines the direction of the output relative to any setpoint error, where:
  - Direct means that the loop output increases when PV increases.
  - Reverse means that the loop output increases with PV decreases.
- Integral gain (K_I): is the value of the property **`Integral Constant`** used to set or reset the gain of the loop expressed in repeats per minute. This value reacts to the duration of the setpoint error.
- Derivative gain: is the value of the property Derivative Constant used to set the derivative or rate gain of the loop expressed in repeats per minute. This value reacts to the rate of change of the setpoint error providing a dampening effect.

Proportional-only control

P-only control is just reset action, where loop output is directly proportional to the magnitude of the setpoint error (E_S) and the size of the proportional gain (K_P).

P-only loop output is linear, and is calculated as follows:

$Output = (K_P \times E_S) + bias$ (if action = direct) or

$Output = -((K_P \times E_S) + bias)$ (if action = reverse)

where $E_S = [PV - setpt]$

If you are using proportional-only loop control, follow these guidelines:

- Output limits: Define the **Maximum Output** and **Minimum Output** properties for the loop output, noting that the maximum value must be greater than the minimum.
- Proportional Gain: Calculate and enter a proportionalConstant (K_P) property value starting with this formula:

  `[output range (maxOutput – minOutput)] / throttling range`

  where `throttling range` is the corresponding result in the process variable.

  For example, for a temperature loop where a 0-to-100% loop output results in a 20 degree swing in the process variable, a starting point KP is:

  `[(100% – 0%) / 20deg.] = [(100% / 20deg.] = 5`

  When tuning the loop, you can try increasing this value (effectively using only a portion of the throttling range) to eliminate the amount of setpoint error. However, if you increase the K_P too much, this typically results in a constant oscillation of the process variable (above and below the setpoint).
- Bias: Assign the bias property an output-midpoint value (for example, 50.0). This allows for equal corrections for a process variable above or below setpoint.
- Integral and Derivitive Gain: Set the properties integralConstant and derivativeConstant to 0.0 (the defaults).

Proportional with Integral (PI) control

PI configuration is recommended for most control loops, because the integral term eliminates the setpoint offset inherent in P-only loops. PI control uses proportional gain to adjust the output, and then incrementally continues to add (or subtract, if appropriate) from the output value for as long as a setpoint error continues to exist.

PI loop output is calculated as follows:

$Output = K_P \times (E_S + K_I \times ErrorSum)$ (if action = direct), or

$Output = -(K_P \times (E_S + K_I \times ErrorSum))$ (if action = reverse)

where: $E_S = [PV - setpt]$

$ErrorSum$ = Sum of E_S over time

Repeats per minute

The **Integral Constant** property specifies the integral gain (K_I) in repeats per minute. This is sometimes called a reset rate. To understand repeats per minute, consider the scenario where a loop is controlling at setpoint. If a certain setpoint error occurs, say from a sudden setpoint change, the loop output immediately changes by a level corresponding to its proportional constant (acting on the P-term). During this hypothetical example, assume the controlled process does not react from any loop output change, but stays at the original value (setpoint error stays constant).

The loop's integral term immediately begins increasing the output (or decreasing the output, depending on the direction of setpoint error) at specific rate determined by the integral term. Over the period of one

minute, the amount of output change that would occur is defined by the **Integral Constant** (repeats per minute). A repeat equals the amount of output change initially generated by the P-term. For example, if this loop was configured with an **Integral Constant** value of 2.0, and the original output change was +7%, over a period of one minute the integral term would linearly ramp up the output value an additional +14%, or two repeats.

In a real-world PI loop, of course, the process variable does respond to an output change, and this continuously-linear ramping of the output would not occur. Instead, the process variable would start moving towards setpoint and the setpoint error would change (changing the proportional and integral terms, thus the loop output).

Integral overshot

The integral term of a PI loop can cause an "overshoot" of setpoint, meaning that the increased loop output may result in a new setpoint error in the opposite direction. In some cases, it is possible for this overshoot to continuously repeat (oscillation), which is typically undesired. However, a small amount of overshoot for an initial correction is not uncommon.

To minimize overshoot, the PI loop's integralConstant is typically kept small, and sized appropriately for the assigned proportionalConstant.

Integral windup prevention

Integral windup is prevented by limiting the ErrorSum value based on the LoopPoint's **Maximum Output** and **Minimum Output** values.

Pi configuration guidelines

If using PI loop control, follow these guidelines:

- Output limits: Define the **Maximum Output** and **Minimum Output** properties for the loop output, noting that the maximum value must be greater than the minimum.
- Proportional gain: Calculate and enter a **Proportional Constant** (K_P) property value starting with this formula:

  `[output range (minOutput – maxOutput)] / throttling range`

  where `throttling range` is the corresponding result in the process variable.

  For example, for a temperature loop where a 0-to-100% loop output results in a 20 degree swing in the process variable, a starting point K_P is:

  `[(100% – 0%) / 20deg.] = [(100% / 20deg.] = 5`

  When tuning a PI loop, you typically reduce the **Proportional lConstant** value, because the integral effect on the output will correct setpoint error over time.
- Bias: Assign a value of 0.0 (no output bias). A fixed bias is not desired, because the integral term of the loop effectively creates an adjustable bias, as needed.
- Integral Gain: Set the integral gain (property **Integral Constant**) to a nominal value, typically less than one (1.0). A value of 0.5 is a good starting point for many loops. Decreasing the integral constant makes the loop respond more slowly.
- Deravitive Gain: Disable derivative by setting the **Derivative Constant** property at 0.0 (the default).

Proportional with Integral and Derivative (PID) control

PID loop control can be difficult to tune and (often for this reason) is seldom used. However, in certain cases, PID control may be needed. An example is the control of a process with a long reaction time, such as temperature control of a large mass. For such a lag-oriented system, the derivative component of the PID loop output can help prevent overshoot that might otherwise result from PI control.

The derivative gain (K_D) exerts an anticipating braking effect on the loop output, based on the rate-of-change of the process.

PID loop output is calculated as follows:

$Output = K_P \times (E_S + K_I \times ErrorSum + K_D \times ((E_S - LastE_S) / deltaT))$ if action = direct) or

$Output = -(K_P \times [E_S + K_I \times ErrorSum) + K_D \times ((E_S - LastE_S) / deltaT)))$ (if action = reverse)

where: $E_S = [PV - setpt]$

$ErrorSum$ = Sum of E_S over time

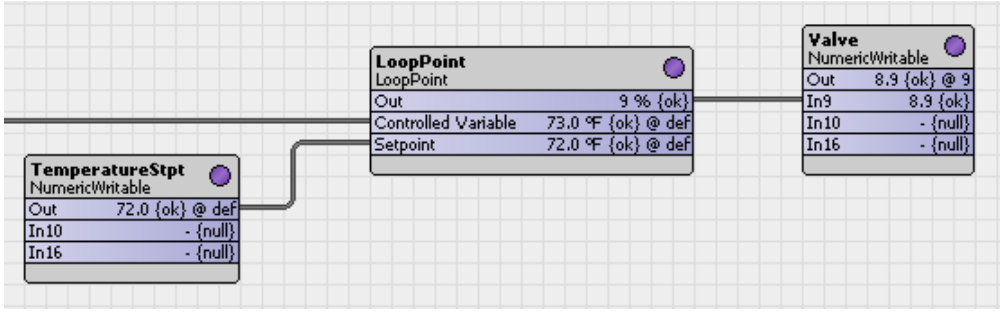
$deltaT$ = time between samples

In the LoopPoint, the **Derivative Constant** property specifies the derivative gain (K_D) directly in seconds (note this differs from some systems using derivative in minutes).

If using PID control, follow the same PI configuration as for Proportional with Integral (PI) control, with the addition of defining a positive value as the **Derivative Constant**. In general, a **Derivative Constant** less than 10 seconds should be tried first, and only then increased (if necessary), providing that the loop output remains stable at steady-state conditions.

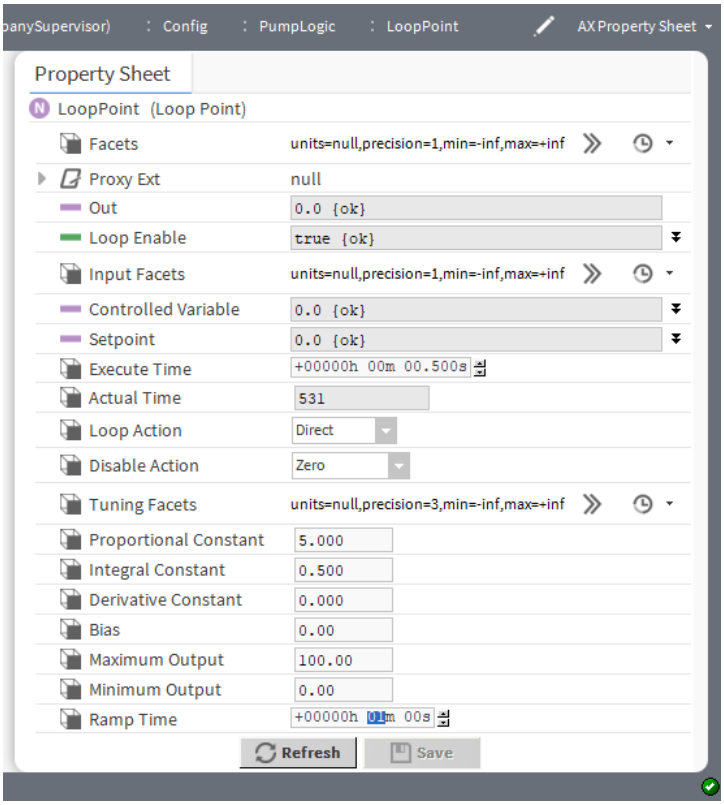
Following is an example of a wire sheet for a loop point.

Figure 9 Example LoopPoint in a wire sheet, linked to other components



The following is the property sheet for the above example.

Figure 10 Example LoopPoint property sheet



Night time heat configuration

At night the air supply to a building needs to provide the least amount of heat. The NightPurge kitControl component configures air supply to meet this requirement.

Two wire sheets demonstrate how the night time purge works.

Figure 11 NightPurge application: purge enabled and active

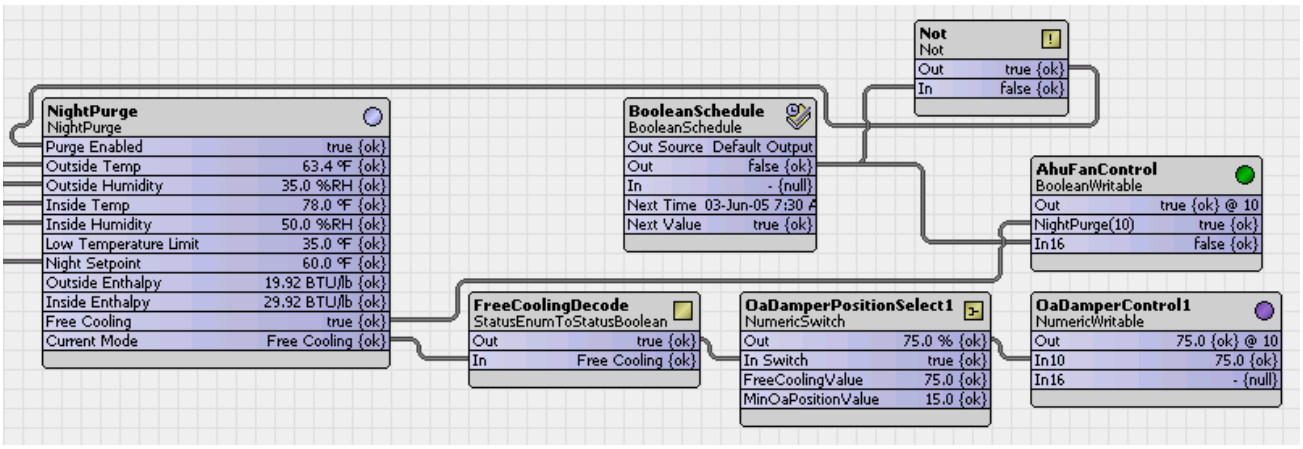
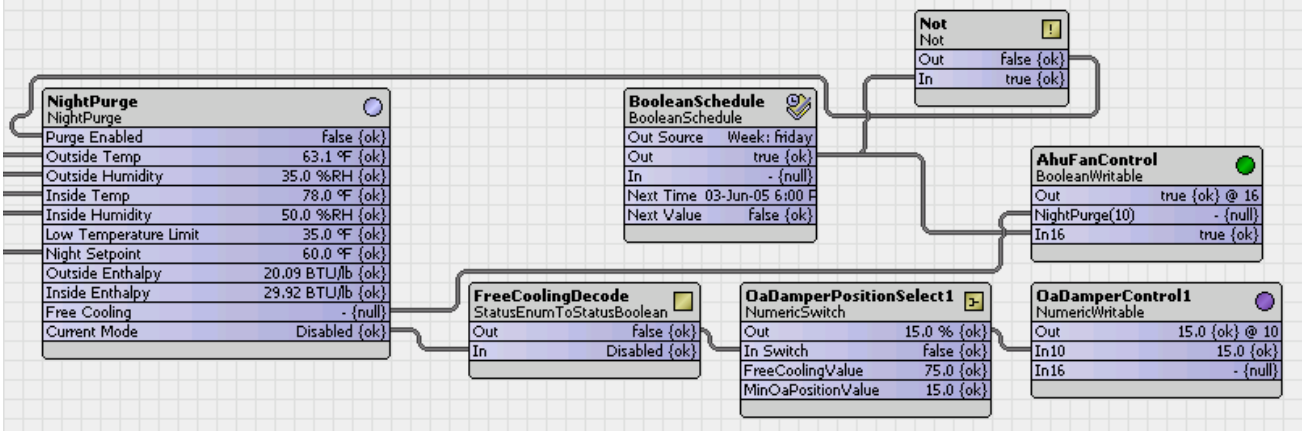
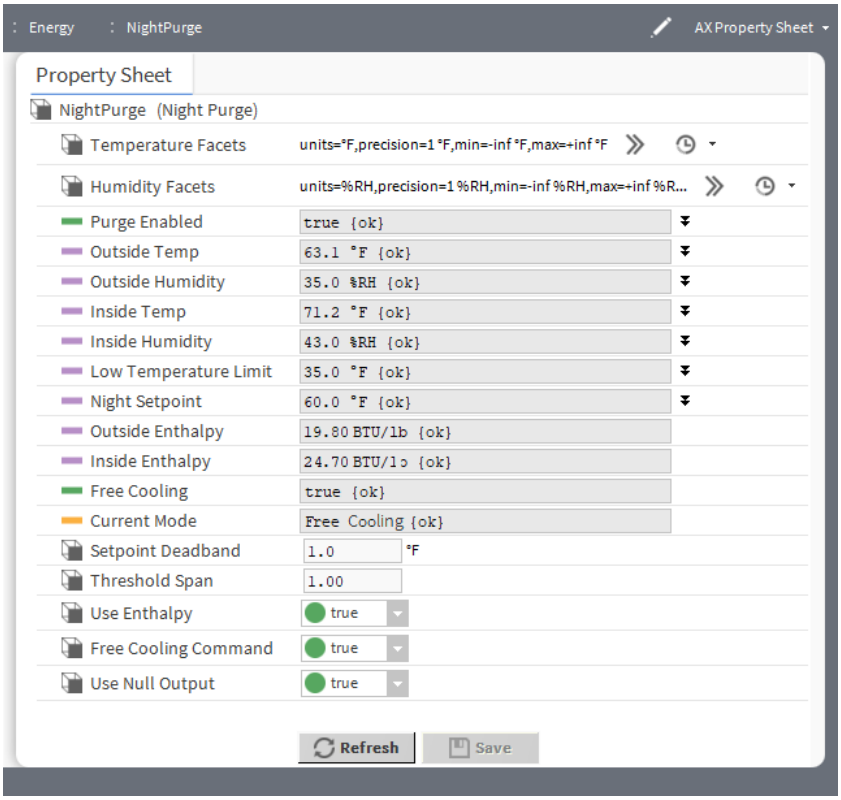


Figure 12 NightPurge application: purge disabled



The following is an example of a NightPurge property sheet.

Figure 13 NightPurge property sheet

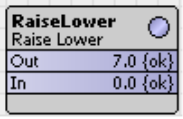


Reversible actuator control

A typical application for the RaiseLower component is to control a reversible actuator to drive a coupled valve or damper either open or closed.

The external hardware device consists of two on-board relays with volt free contacts that switch to provide power to either the open or the close command of the actuator. The component activates either relay for a proportion of the full scale drive time of the actuator (drive time is pre-defined by the manufacturer).

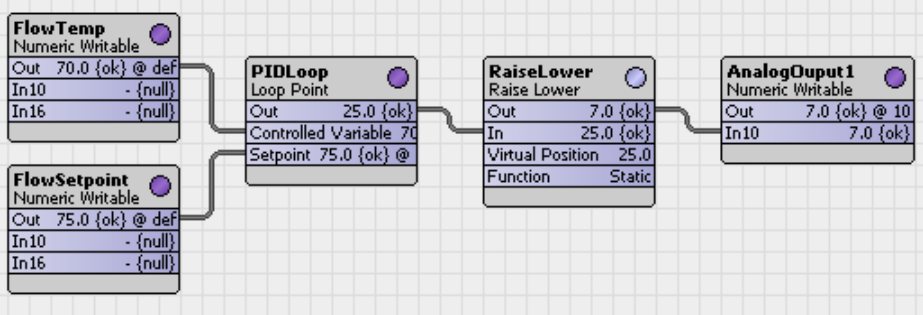
Figure 14 RaiseLower component



The acceptable input to this device is 0 to 10 volts, with staged control at 0v, 4v, 7v and 10v as detailed below:

Volts	Raise	Lower	Function
0	Off	Off	Off
4	Off	On	Lower
7	Off	Off	Static
10	On	Off	Raise

Figure 15 Example using the RaiseLower component



The wire sheet is an example of an actuator configured for a 100 second full-scale drive time. The initial actuator and virtual position values of are set at 0% (synchronized at fully lowered). Should the input signal increase to 40 percent, the raise output turns on for 40 percent of the full scale drive time (40 sec). A subsequent input decrease to 15 percent lowers the output to 25 percent of full scale drive time (25 sec) moving the actuator to 15 percent open.

Heating and cooling management to reduce energy consumption

In a simple model, a schedule can control when the heat or air conditioning turn on and off. Using the OptimizedStartStop kitControl component you can configure a more sophisticated automatic system that optimizes heating and cooling start and stop times based on actual outside and inside temperatures.

To conserve energy in a building, the OptimizedStartStop kitControl component serves two goals:

- Turn the heating or cooling equipment on at the latest possible time while still providing a comfortable temperature during building occupancy.
- Turn the same equipment off at the earliest possible time, allowing the building to "drift" while staying within a comfortable temperature range until it is no longer occupied.

Starting and stopping heating or cooling equipment are valid schedule events, which you link to the OptimizedStartStop component. Setting the component's **`Start Enable`** or **`Stop Enable`** property to `true` initiates the heating or cooling event on behalf of the schedule. Once started or stopped, the component runs a calculation 15 seconds after the beginning of every minute until the next event, which is *not already set* in the schedule. The result of the calculation is called the **`Calculated Command Time`**. The value of this property determines the next scheduled event: an early start command to achieve a specified temperature range by the time the building is occupied or an early stop command without sacrificing the temperature range by the time the space is unoccupied. After a schedule invokes the **`Calculated Command Time`**, the component

analyzes the actual area response (temperature change rate) and makes weighted adjustments to the calculation parameters based on the detected values so that subsequent calculations might be more accurate.

If one of the enable values (**`Start Enable`** or `Stop Enable`) is scheduled to be set to `true` in one hour, but is already `true`, the component performs no calculation even if **`Start Enable`** or **`Stop Enable`** is set to true.

Calculated start and stop times

The OptimizedStartStop component optimizes only one start sequence per day. You can perform multiple stop operations but no optimized stop can occur before the time specified by the **`Earliest Stop Time`** property.

These factors define the calculated start time:

- **Temperature differential**

  If the inside (space) temperature is outside the range defined by the lower and upper comfort limits, the difference between the space temperature and the closer limit represents the number of degrees the mechanical equipment must make up during the prestart (optimized) period.

- **Run-time minutes**

  Whether to heat or cool depends on the direction that the space temperature must move. The component multiplies this run-time value by the temperature differential to determine the number of run-time minutes required to achieve the comfort limit at occupancy time, as defined by the schedule's start time.

- **Optimum start time**

  This is the best time to start heating or cooling before the scheduled start for the day. If the system's time is later than the schedule's time offset by the calculated lead time, the component enables optimum start outputs.

  NOTE: If the calculated lead time is so large that an optimum start time prior to midnight is the result, the optimum start occurs at midnight. The component performs an optimum start only before the first scheduled start for the day.

Similar factors define stop time:

- **Temperature differential**

  If the inside (space) temperature is inside the range defined by the lower and upper comfort limits and if the schedule's status is active, the difference between the space temperature and one of the limits (depending on the mode) represents the number of degrees the temperature can drift between the time the mechanical equipment is stopped and the schedule's inactive event time.

- **Drift time**

  The drift (lead time) calculation is similar to the one for start time but using the drift-time heating and cooling factors.

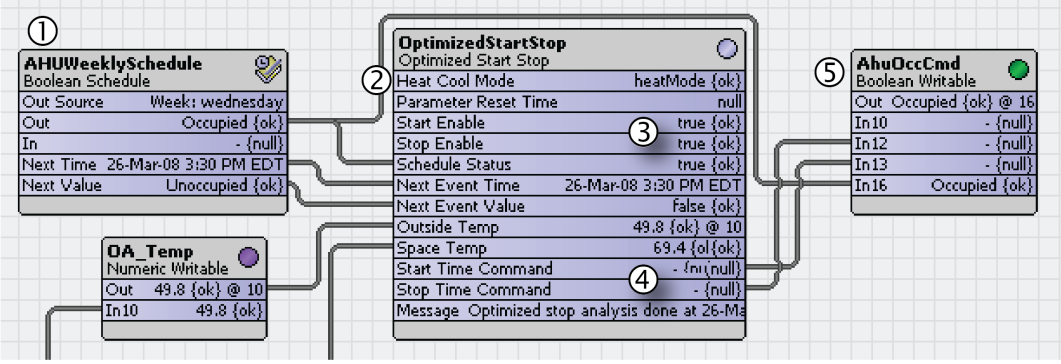
- **Optimum stop time**

  The component invokes the optimum stop time for each of the schedule's inactive events based on the drift time and **`Next Event Time`** value.

Optimized start example

To optimize performance, the framework turns an air handling unit on and off based on a weekly schedule.

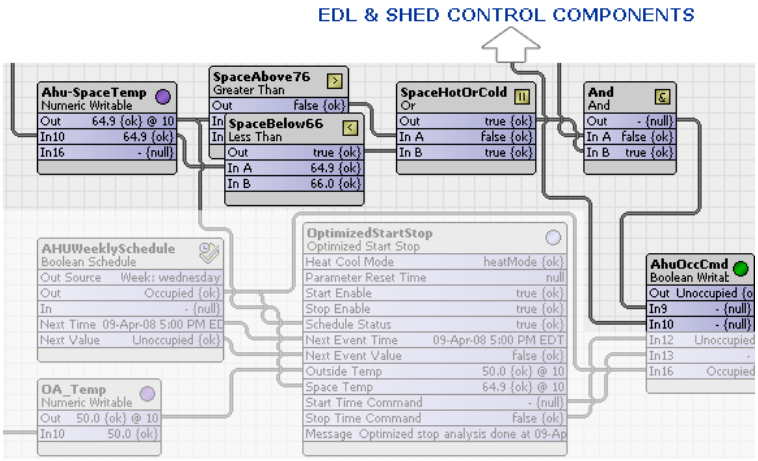
Figure 16 A simple OptimizedStartStop component wire sheet



In this example:

- A weekly schedule specifies occupancy times and is linked to the component for calculations (1). The schedule is also linked directly (bypassing the component) to the **In16** property (lowest priority in this case) of the occupancy control point (AhuOccCmd) (2).
- The components **Start Enable** and **Stop Enable** properties (3) are both `true`. This means that both optimized starts and optimized stops are enabled.
- The **Start Time** and **Stop Time Commands** (4) are linked to priority inputs **In12** and **In13** respectively of the occupancy control point (AhuOccCmd) (5). These commands are true when an optimal start or stop condition is required, otherwise the outputs are set to `null`, which relinquishes control to the next higher priority level.

Figure 17 Another OptimizedStartStop component wire sheet



Additional logic is linked into the occupancy command component (AhuOccCmd) to control which logic has priority on specifying the AhuOccCmd Boolean point status, as follows:

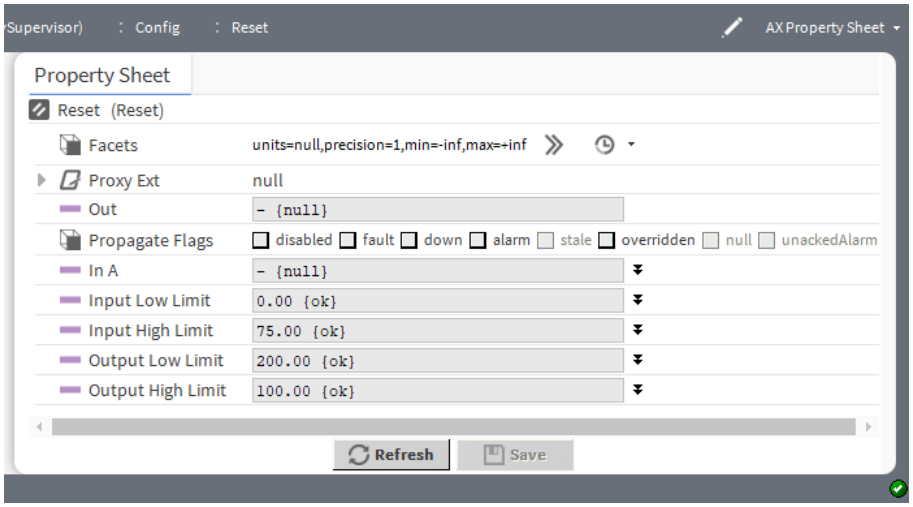
- **In9** temperature control overrides a demand limiting link from an EDL component to **In10**. This prevents a load shed if the comfort range that can be configured is exceeded.
- **In10** (demand limiting link from the EDL component) overrides an OptimizedStartStop component's Stop link into **In12**.
- **In12** (OptimizedStartStop component's Stop link) overrides a Start link into **In13** (as described above).
- **In13** (optimal start) overrides the schedule link to **In16** (lowest priority) (as described above).

Hot water control setpoint based on outside air temperature

A Reset math object establishes a hot water control setpoint based on the outside air temperature at **`InA`**.

The Reset component is located in the **Math** folder of the **`kitControl`** palette.

Figure 18 Reset properties



When the outside air temperature is 0° F, the hot water setpoint is 200° F. When the outside air temperature is 75° F, the hot water setpoint is 100° F. The Reset object is configured as:

- `Input Low Limit = 0.0`
- `Input High Limit = 75.0`
- `Output Low Limit = 200.0`
- `Output High Limit = 100.0`

If the **`InA`** value is beyond the input limits, the output is limited by the corresponding output limit (in this case, 200 at 0° F or below, 100 at 75° F or above). When the input is at an intermediate value, the output scales linearly. For example, when the outside air temperature is at 38.2° F, the Reset output is 149.1° F.

Data gathering

These examples illustrate how to configure Latch components to collect Boolean and numeric data. Each latch component contains a **`Clock`** property and a `Latch` action. Either can trigger the collection of data (capturing the **`In`** value from a control point and making it available as an **`Out`** value).

The examples are similar, in that they all use a Schedule component to invoke a latch. Other components may be used to invoke a latch, however, any latch you invoke using the **`Clock`** property must include a method for setting the **`Clock`** property status back to `false` before the clock is available for latching again. The third example illustrates the use of a latch component's latch action instead of using the **`Clock`** property.

Chapter 3 Alarm components

Topics covered in this chapter

♦ kitControl-AlarmCountToRelay
♦ kitControl-ChangeOfStateCountAlarmExt
♦ kitControl-ElapsedActiveTimeAlarmExt
♦ kitControl-LoopAlarmExt

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. The Alarm folder in the **kitControl** palette contains four special-purpose alarm extensions.

Two extensions are for Boolean type points that also have one or more DiscreteTotalizerExt extensions:

- ChangeOfStateCountAlarmExt
- ElapsedActiveTimeAlarmExt

These extensions provide alarm capability based upon COS count and runtime (elapsed active time). To use either (or both) of these extensions in the parent Boolean point (BooleanPoint, BooleanWritable) place them under the corresponding DiscreteTotalizerExt.

The LoopAlarmExt provides a sliding-limit type alarm for a LoopPoint. Its calculations are based on control deviation from a setpoint.

You can configure the AlarmCountToRelay extension to monitor the number of alarms (alarm count) of a linked Alarm Class component. AlarmCountToRelay features a timed Boolean relay output.

An additional alarm extension, StringChangeOfValueExt is in the **alarm** palette. This extension generates an alarm upon inclusion or exclusion of a particular string value, or more accurately, regular expression (regexp) of a point's Out slot (string type).

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

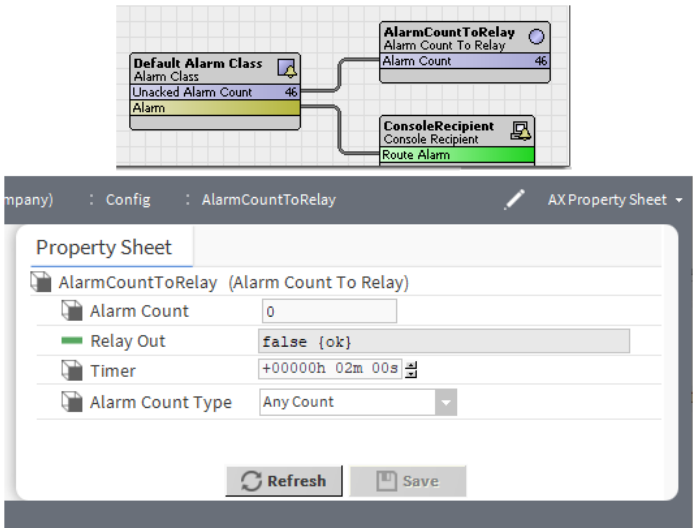
kitControl-AlarmCountToRelay

This component links from an Alarm Class component to monitor alarm count and send an associated Boolean output to a relay when the alarm count increases.

AlarmCountToRelay is in the **Alarm** folder of the **kitControl** palette.

The alarm count type to monitor is optional and includes the alarm states or statuses. Use this component in security applications or situations to connect the occurrence of a new alarm with an event, such as a light, horn, or other signal, which requires a relay connection.

Figure 19 AlarmCountToRelay property sheet



The screen capture shows an AlarmCountToRelay component that is linked to the **`Unacked Alarm Count`** property. In this example, the unacknowledged Alarm Count is 46. Any increase in this number causes the **`Relay Out`** property to change status from `false` to `true` for a time equal to the value of the **`Timer`** property.

Property	Value	Description
Alarm Count	number	Displays the current alarm count for an alarm class component that is linked to this component. The numeric value of this property dynamically displays the number of alarms of the type specified in the **`Alarm Type Count`** property.
Relay Out	read-only `true` or `false {ok}` (default)	Provides a Boolean output value for linking into a relay control component. `false` indicates no change in the alarm count. `true` indicates a change in the Alarm Count (alarm count is active). When an Alarm Count increases, the framework changes this property from `false` to `true` and maintains this status for the amount of time specified by the **`Timer`** property value.
Timer	hours minutes seconds (defaults to two minutes)	Defines how long to hold the **`Relay Out`** in the active (`true`) state.
Alarm Count Type	drop-down list	Selects the type of alarm for which to monitor the count. `Any Count` links from any type of alarm to the Alarm Count property. `Unacked Alarm Count` links from the **`Unacked Alarm Count`** property on an Alarm Class component to the **`Alarm Count`** property of this component. `Open Alarm Count` links from the **`Open Alarm Count`** property on an Alarm Class component to the **`Alarm Count`** property of this component. In Alarm Count links from the **`In Alarm Count`** property on an Alarm Class component to the **`Alarm Count`** property of this component.

kitControl-ChangeOfStateCountAlarmExt

This special-purpose alarm extension is especially for use as a child of a BooleanPoint or BooleanWritable that has one or more DiscreteTotalizerExt extensions. It provides alarming on COS (change of state) counts, using the offNormal property errorLimit.

ChangeOfStateCountAlarmExt is available in the **Alarm** folder of the **kitControl** palette, along with a ElapsedActiveTimeAlarmExt (for runtime-based alarms). Both extensions can reference the same DiscreteTotalizerExt.

In the parent Boolean point, order the ChangeOfStateCountAlarmExt slot below the DiscreteTotalizerExt slot that it references. In the ChangeOfStateCountAlarmExt's Offnormal container, use the **Discrete Totalizer Select** property to reference the DiscreteTotalizerExt.

The properties for this extension are the same as those for a standard framework alarm extension. Refer to the .

kitControl-ElapsedActiveTimeAlarmExt

This component is a special-purpose alarm extension, which is especially for use as a child of a BooleanPoint or BooleanWritable that has one or more **DiscreteTotalizerExt** extensions. It provides alarming on runtime (elapsed active time), using the offNormal property **errorLimit**.

ElapsedActiveTimeAlarmExt is available in the**Alarm** folder of the **kitControl** palette along with a ChangeOfstateCountAlarmExt extension for COS-based alarms. You can use both extensions to reference the same DiscreteTotalizerExt.

In the parent Boolean point, order the ElapsedActiveTimeAlarmExt slot below the DiscreteTotalizerExt slot that it references. In the ElapsedActiveTimeAlarmExt's Offnormal container, use the **Discrete Totalizer Select** property to reference the DiscreteTotalizerExt.

The properties for this extension are the same as those for a standard framework alarm extension. Refer to the *Niagara Alarms Guide.*

kitControl-LoopAlarmExt

This component provides alarming as a deviation-from-current-setpoint (plus or minus), using offnormal properties errorLimit and deadband. It is a special-purpose alarm extension, especially for use as child of a LoopPoint component.

LoopAlarmExt is available in the **Alarm** folder of the **kitControl** palette.

The properties for this extension are the same as those for a standard framework alarm extension. Refer to the *Niagara Alarms Guide.*

Chapter 4 Constant components

Topics covered in this chapter

♦ kitControl-BooleanConst
♦ kitControl-EnumConst
♦ kitControl-NumericConst
♦ kitControl-StringConst

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. Each of these components provides a linkable status-type output value, and offers a default action for changing its value.

A constant may be handy when the same target property of multiple components needs to be adjusted simultaneously. Constants may also help to clarify logic on a wire sheet.

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

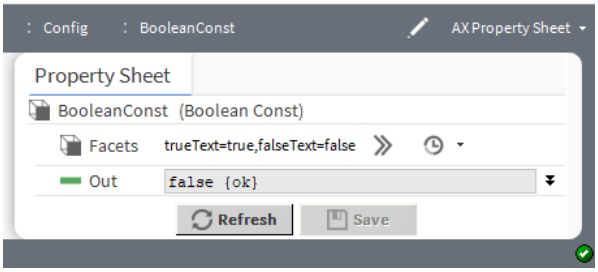
kitControl-BooleanConst

This component provides a constant status Boolean value with two actions to set.

BooleanConst is available in the **Constants** folder of the **kitControl** palette.

It provides a linkable status-type output value, and two actions for changing its value: Active (true) and Inactive (false).

Figure 20 BooleanConst properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines a range of enumerated states (Boolean values).
Out	StatusBoolean `true` and `false` (default) and null definition	Shows the current binary state of the point (on or off).

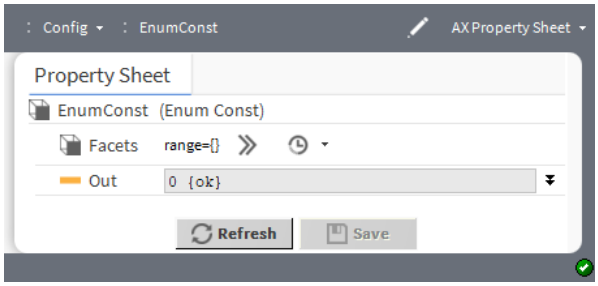
kitControl-EnumConst

This component provides a constant EnumStatus value, with available action to Set.

EnumConst is available in the **Constants** folder of the **kitControl** palette.

It provides a linkable status-type output value, and two actions for changing its value: Active (true) and Inactive (false). A constant may be handy when the same target property of multiple components needs to be adjusted simultaneously. Constants may also help to clarify logic on a wire sheet.

Figure 21 EnumConst properties



Property	Value	Description
Facets	Config Facets window (defaults to zero (0))	Defines a range of enumerated states (Enum values).
Out	enumerated list (defaults to zero (0), that is, no list)	Shows the current enum value associated with the point.

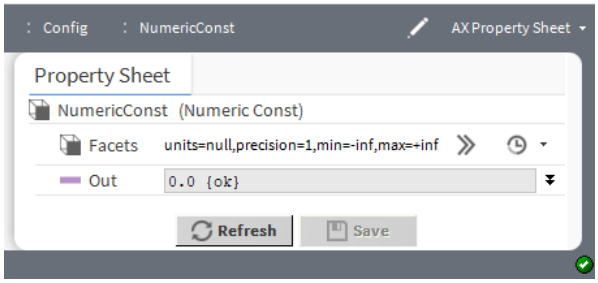
kitControl-NumericConst

This component provides constant StatusNumeric value, with available action to Set.

NumericConst is available in the **Constants** folder of the **kitControl** palette

Constant components are far simpler than writable control points (no priorities, status processing, and so forth). Constant components provide a linkable status-type output value and a default Set action for changing that value. This type of object may be handy when the same target property of multiple components needs to be adjusted simultaneously. Constants may also help clarify logic on a wire sheet.

Figure 22 NumericConst properties



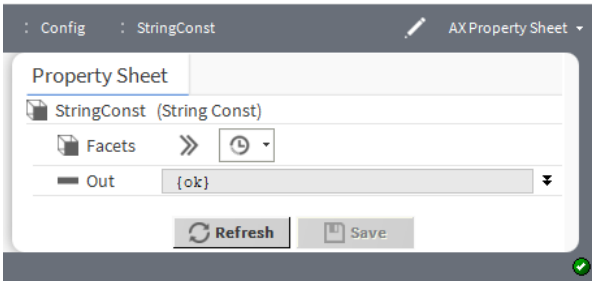
Property	Value	Description
Facets	Config Facets window (defaults to null for units, one decimal place for precision, negative infinity for minimum, and positive infinity for maximum)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed.
Out	number (defaults to zero (0))	Shows the current binary state of the point (on or off).

kitControl-StringConst

This component Provides constant StatusString value, with available action to Set.

StringConst is available in the **Constants** folder of the **`kitControl`** palette.

Figure 23 StringConst properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Associates a key with one or more phrases.
Out	text (defaults to blank) and null definition	Displays the phrase associated with the point.

Chapter 5 Conversion components

Topics covered in this chapter

- kitControl-BooleanToStatusBoolean
- kitControl-DoubleToStatusNumeric
- kitControl-EnumToStatusEnum
- kitControl-FloatToStatusNumeric
- kitcontrol-IntToStatusNumeric
- kitControl-LongToStatusNumeric
- kitControl-StringToStatusString
- kitControl-StatusBooleanToBoolean
- kitControl-StatusEnumToEnum
- kitControl-StatusEnumToInt
- kitControl-StatusEnumToStatusBoolean
- kitControl-StatusEnumToStatusNumeric
- kitControl-StatusNumericToStatusEnum
- kitControl-StatusNumericToDouble
- kitControl-StatusNumericToFloat
- kitControl-StatusNumericToInt
- kitControl-StatusStringToStatusNumeric
- kitControl-StatusNumericToStatusString
- kitControl-NumericUnitConverter

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. In most cases, a conversion object takes an input value and makes it available in the object's output as a different type of data. Prior to AX-3.6, they typically served to link different data types. This functionality is now standard, which means that for most applications you do not need a conversion component. Instead, the framework automatically converts data type between linked components.

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

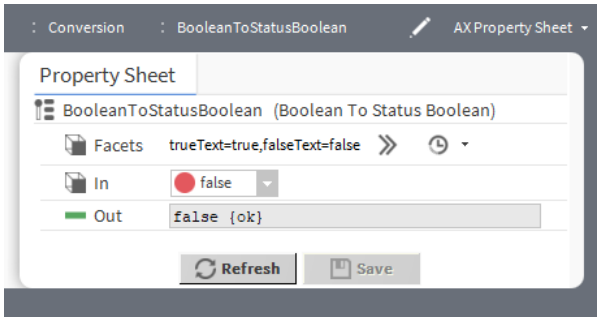
- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

kitControl-BooleanToStatusBoolean

This component links (converts) the simple Boolean value (a non-status value) in an extension or other component (**`In`** property) to the **`Out`** property of a point or kitControl object (a StatusBoolean value). This component is the reverse of the StatusBooleanToBoolean component.

BooleanToStatusBoolean is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 24 BooleanToStatusBoolean properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`. This definition applies to the **`In`** property value. What text to define depends on how you are using the component.
In	`true` and `false` (default)	Displays the source simple non-status value.
Out	read-only StatusBoolean `true` and `false`	Displays the converted target status value.

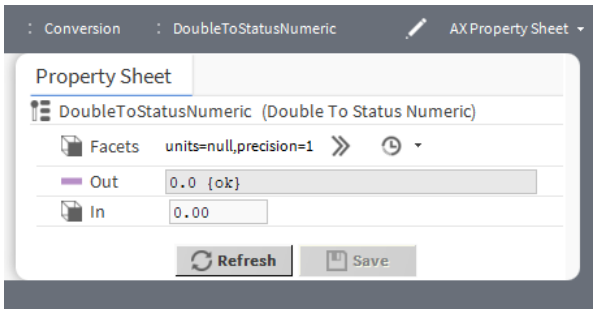
kitControl-DoubleToStatusNumeric

This component converts a double value to StatusNumeric.

DoubleToStatusNumeric is available in the **Conversion** folder of the **`kitControl`** palette.

The conversion object is often used in a AX-3.5 or earlier host. This component links from the source non-status value of a property in an extension or other component to the target **`In`** property of a point or kitControl object (status value).

Figure 25 DoubleToStatusNumeric properties



Property	Value	Description
Facets	Config Facets window (defaults to null for units and one decimal place of precision)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This definition applies to the **`In`** value for this component.
Out	read-only number	Displays the converted status numeric value.
In	number (defaults to zero (0))	Displays the source double value.

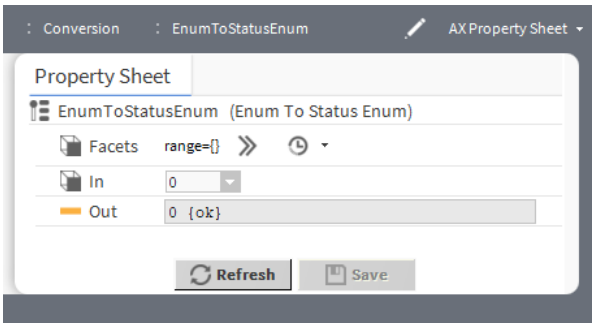
kitControl-EnumToStatusEnum

This component converts an Enum value to StatusEnum, one of multiple enumerated states in the operating range of a component.

EnumToStatusEnum is available in the **Conversion** folder of the **`kitControl`** palette.

The conversion object is often used in a AX-3.5 or earlier host. This component links from the source non-status value of a property in an extension or other component to the target **`In`** property of a point or kitControl object (range of StatusEnum values).

Figure 26 EnumToStatusEnum properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines a range of enumerated states (enum values) This value applies to the **`In`** property of this component.
In	number (defaults o zero (0))	Displays the source non-status enum value.
Out	read-only enumerated value	Displays the converted target status enum value.

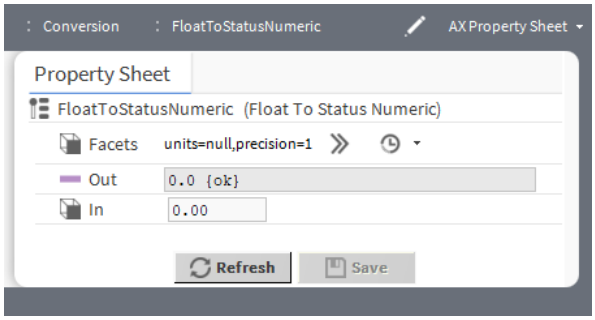
kitControl-FloatToStatusNumeric

This component converts a Float value to a StatusNumeric.

FloatToStatusNumeric is available in the **Conversion** folder of the **`kitControl`** palette.

The conversion object is often used in a AX-3.5 or earlier host. This component links from the source non-status **`Out`** value of a property in an extension or other component to the target **`In`** property of a point or kitControl object (StatusNumeric value).

Figure 27 FloatToStatusNumeric properties



Property	Value	Description
Facets	Config Facets window (defaults to null and a single decimal position)	Selects units and configures how the values displays: `units` define the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target status numeric value.
In	number	Displays the source float value.

kitcontrol-IntToStatusNumeric

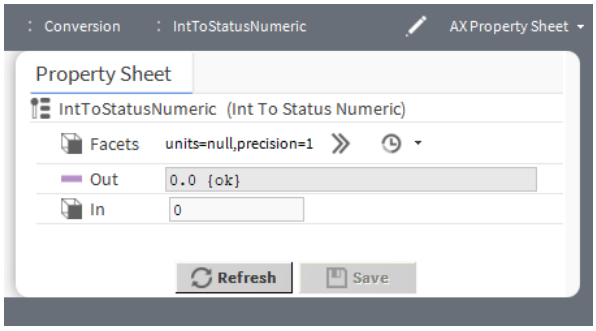
This component converts an Int (integer) value to StatusNumeric.

IntToStatusNumeric is in the **Conversion** folder of the **`kitControl`** palette.

The conversion object is often used in a AX-3.5 or earlier host. This component links from the source integer value of a property in an extension or other component to the target **`In`** property of a point or kitControl object (StatusNumeric value).

For example, to link the **`changeOfStateCount`** property of a BooleanPoint's DiscreteTotalizer extension to a math object input, you would link an IntToStatusNumeric between the two.

Figure 28 IntToStatusNumeric properties



Property	Value	Description
Facets	Config Facets window (defaults to null for units and one decimal place for precision)	Selects units and configures how the values display: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target status numeric value.
In	number	Displays the source integer value.

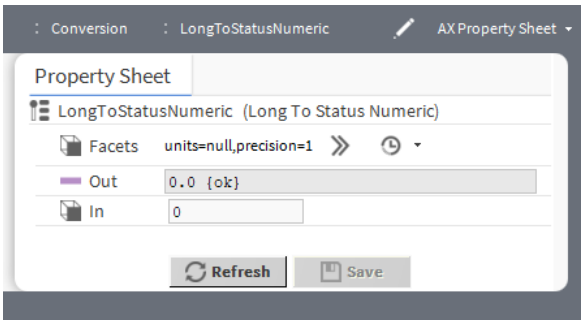
kitControl-LongToStatusNumeric

This component converts a Long value to StatusNumeric.

LongToStatusNumeric is available in the Conversion folder of the kitControl palette.

The conversion object is often used in a AX-3.5 or earlier host. This component links from the source non-status **`Out`** value of a property in an extension or other component to the target **`In`** property of a point or kitControl object (StatusNumeric value).

Figure 29 LongToStatusNumeric properties



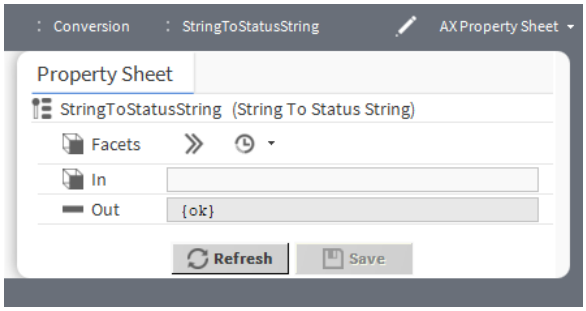
Property	Value	Description
Facets	Config Facets window (defaults to null for units and a single decimal place)	Selects unit and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target status numeric value.
In	number	Displays the source long value.

kitControl-StringToStatusString

This component converts the string **`In`** value to a StatusString **`Out`** value.

StringToStatusString is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 30 StringToStatusString properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Associates a key with one or more phrases.
In	text	Displays the source simple phrase (text string).
Out	read-only text	Displays the converted target phrase (status string).

kitControl-StatusBooleanToBoolean

This component converts a StatusBoolean **In** value to a simple Boolean **Out** value. This component is the reverse of the BooleanToStatusBoolean component.

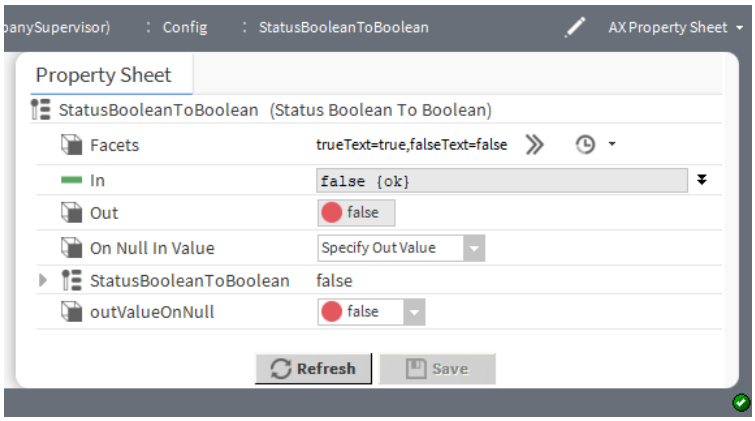
StatusBooleanToBoolean is available in the **Conversion** folder of the **kitControl** palette.

This component converts an units output from a source component to the units of the target component. Such conversion is automatic. But, if needed, you can use this component to convert between:

- Source **Out** of a point, kitControl, or Schedule object (status value), to a
- Target non-status **In** value of a property in an extension or other component.

For example, to link the **Out** of a Schedule object to the **Enabled** property of a history extension of a point or object, you link a StatusBooleanToBoolean object between the two.

Figure 31 StatusBooleanToBoolean properties



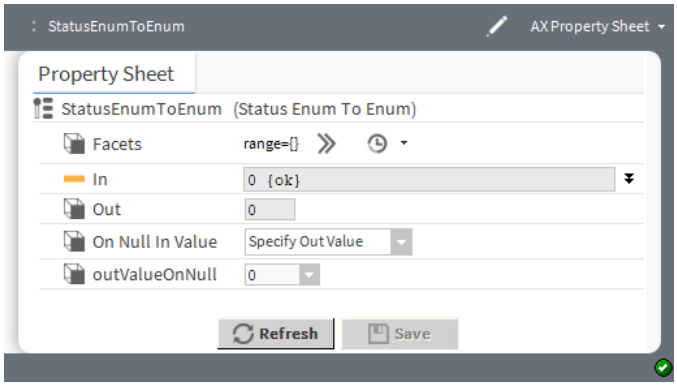
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns true. `falseText` configures the text to describe the state when the component returns false. This definition applies to the **`In`** property value. What text to define depends on how you are using the component.
In	`true` and `false` (default)	Displays the source status Boolean value.
Out	read-only StatusBoolean `true` and `false`	Displays the converted target simple Boolean value.
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **`In`** default value when the result of the conversion is null. Normally, the **`Fallback`** property for a given point defines the meaning of null as 0.0. If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null value for the point. For example, a NumericWritable point (proxy point) positions a damper from 0 to 100% open. In the point's configuration, its **`Fallback`** property defaults to a null value of 0.0. If this null status is unchecked and another value, such as 20, is entered and saved, the new value becomes the default when the value for the point is null. If this modified component links to another StatusNumeric property, the framework ignores the null value. But, if connected to a StatusBooleanToBoolean, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **`Out Value On Null`** property to this component. Configure it to return a reasonable value when the **`In`** value is null.
Out Value On Null	number	Defines an **`Out`** value for the conversion object when its **`In`** value has a null status.

kitControl-StatusEnumToEnum

This component converts a StatusEnum **`In`** value to a simple Enum **`Out`** value.

StatusEnumToEnum is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 32 StatusEnumToEnum properties



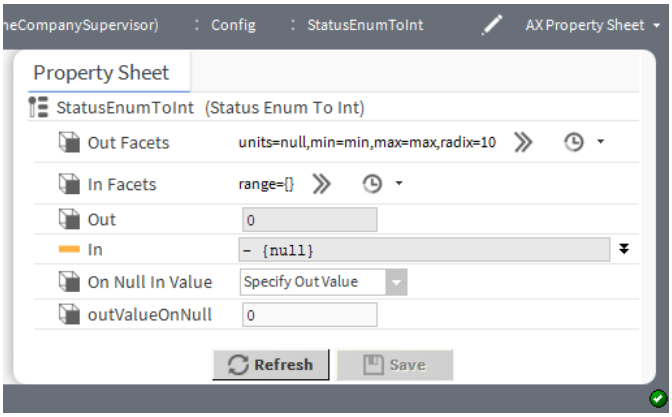
Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines a range of enumerated states(enum values). This configuration applies to the **Out** property value.
In	number (defaults to 0) and null definition	Displays the source status enum value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**In** value.
Out	read-only enumerated value	Displays the converted target simple enum value.
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **In** default value when the result of the conversion is null. Normally, the **Fallback** property for a given point defines the meaning of an Enum null as zero (0). If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null default value for the point. If this modified component links to another StatusEnum property, the framework ignores the null value. But, if connected to a StatusEnumToEnum component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **outValueOnNull** property to this component. Configure it to return a reasonable value when the **In** value is null.
Out Value On Null	number	Defines an **Out** value for the conversion object when its **In** value has a null status.

kitControl-StatusEnumToInt

This component converts a StatusEnum **Out** value to an Integer **In** value.

StatusEnumToInt is available in the **Conversion** folder of the **kitControl** palette.

Figure 33 StatusEnumToInt properties



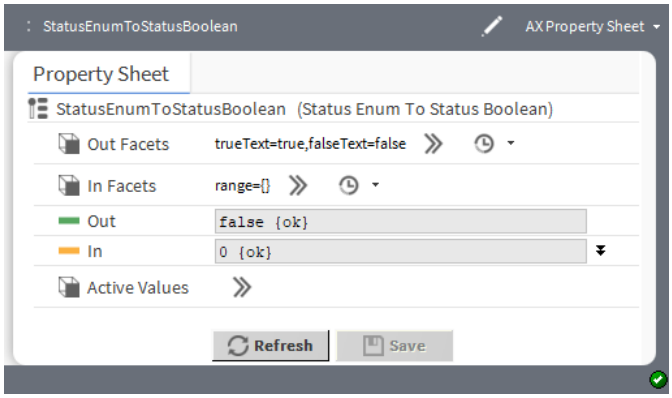
Property	Value	Description
Out Facets	Config Facets window (defaults to null for units, min for minimum, max for maximum and 10 for radix)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `min` defines the minimum value. `max` defines the maximum value. `radix` defines the base of the numeration system. This configuration applies to the **`Out`** property value.
In Facets	Config Facets window	Defines the range of enumerated states (enum values) for the **`In`** value.
Out	read-only integer	Displays the target integer.
In	number (defaults to 0) with null definition	Displays the source status enum value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`**value.
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **`Out`** default value when the result of the conversion is null. Normally, the **`Fallback`** property for a given point defines the meaning of an Enum null as zero (0). If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null default value for the point. If this modified component links to another StatusEnum property, the framework ignores the null value. But, if connected to a StatusEnumToInt component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **`outValueOnNull`** property to this component. Configure it to return a reasonable value when the **`In`** value is null.
Out Value On Null	number	Defines an **`In`** value for the conversion object when the input **`Out`** value has a null status.

kitControl-StatusEnumToStatusBoolean

This component converts a StatusEnum **Out** value to a StatusBoolean **In** value.

StatusEnumToStatusBoolean is available in the **Conversion** folder of the **kitControl** palette

Figure 34 StatusEnumToStatusBoolean properties



Property	Value	Description
Out Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when components returns `false`. This configuration applies to the **Out** property value.
In Facets	Config Facets window	Defines a range of enumerated states (enum values). This configuration applies to the **In** property value.
Out	read-only StatusBoolean `true` and `false`	Displays the target converted status Boolean value.
In	enumerated value (defaults to 0, that is, no value)	Displays the source status enum value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

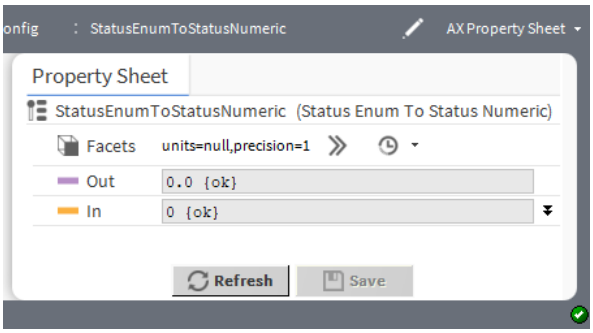
Property	Value	Description
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **`Out`** default value when the result of the conversion is null. Normally, the **`Fallback`** property for a given point defines the meaning of an Enum null as zero (0). If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new default null value for the point. If this modified component links to another StatusEnum property, the framework ignores the null value. But, if connected to a StatusEnumToStatusBoolean component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **`outValueOnNull`** property to this component. Configure it to return a reasonable **`In`** value when the **`Out`** value is null.
Out Value On Null	number	Defines an **`In`** value for the conversion object when the input **`Out`** value has a null status.

kitControl-StatusEnumToStatusNumeric

This component converts a StatusEnum **`Out`** value to a StatusNumeric **`In`** value.

StatusEnumToStatusNumeric is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 35 StatusEnumToStatusNumeric properties



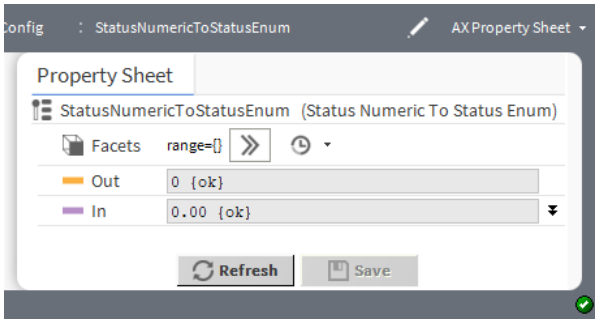
Property	Value	Description
Facets	Config Facets window (defaults to null for units and one decimal place of precision)	Selects units and configures how the value displays: `units` defines the measure from `acceleration` to the `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target status numeric value.
In	number and null definition (defaults to 0)	Displays the source status enum value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.

kitControl-StatusNumericToStatusEnum

This component converts a StatusNumeric value to a StatusEnum value.

StatusNumericToStatusEnum is available in the **Coversion** folder of the **kitControl** palette.

Figure 36 StatusNumericToStatusEnum



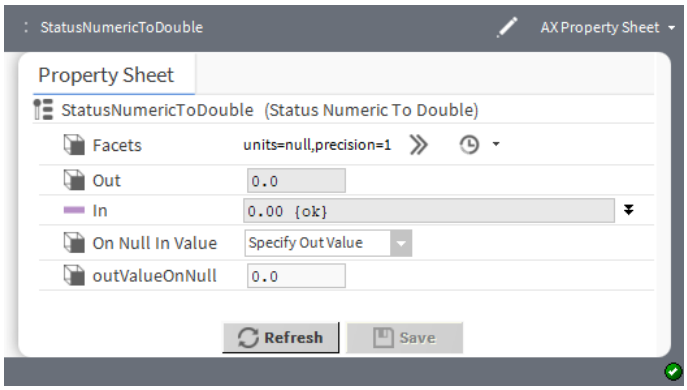
Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines a range of enumerated states (enum values). This configuration applies to the **In** property value.
Out	read-only enumerated value	Displays the converted target status enum value.
In	number and null definition (defaults to 0)	Displays the source status numeric value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

kitControl-StatusNumericToDouble

This component onverts a StatusNumeric value to a Double value.

StatusNumericToDouble is available in the **Conversion** folder of the **kitControl** palette.

Figure 37 StatusNumericToDouble properties



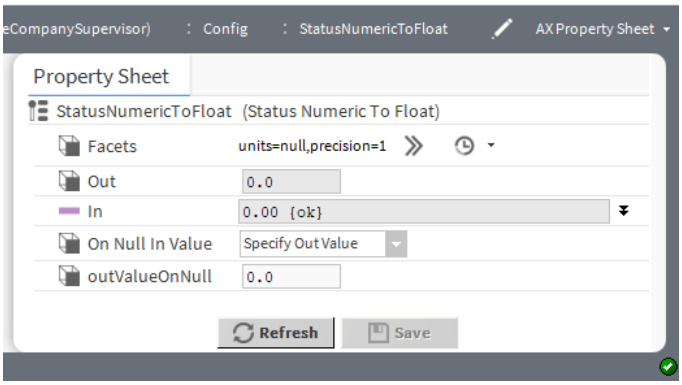
Property	Value	Description
Facets	Config Facets window (defaults to units=null, precision=1)	Selects units and configures how the value displays: `units` defines the unit of measure of `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the target double value.
In	number and null definition (defaults to 0)	Displays the source status numeric value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

kitControl-StatusNumericToFloat

This component converts a StatusNumeric value to a Float value.

StatusNumerictoFloat is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 38 StatusNumericToFloat properties



Property	Value	Description
Facets	Config Facets window (defaults to null for units and one decimal place of precision)	Selects unit and configures how the value displays: `units` define the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target float value.
In	number and null definition (defaults to 0)	Displays the source status numeric value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

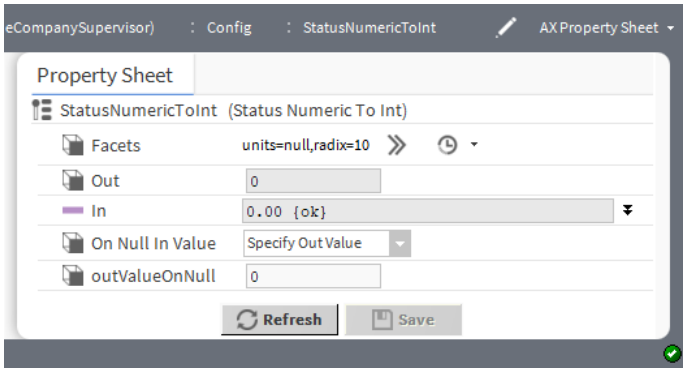
Property	Value	Description
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **`Out`** default value when the result of the conversion is null. Normally, the **`Fallback`** property for a given point defines the meaning of null as 0.0. If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null value for the point. If this modified component links to another StatusNumeric property, the framework ignores the null value. But, if connected to a StatusNumericToFloat component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **`Out Value On Null`** property to this component. Configure it to return a reasonable value when the **`Out`** value is null.
Out Value On Null	number	Defines an **`In`** value for the conversion object when its **`Out`** value has a null status.

kitControl-StatusNumericToInt

This component converts a StatusNumeric value to an Int (integer).

StatusNumericToInt is available in the **Conversion** folder of the **`kitControl`** palette.

Figure 39 StatusNumericToInt properties



Property	Value	Description
Facets	Config Facets window	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `radix` defines the base of the numeration system. This configuration applies to the **`In`** property value.
Out	read-only integer	Displays the converted target integer.
In	number and null definition (defaults to 0)	Displays the source status numeric value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

Property	Value	Description
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **`Out`** default value when the result of the conversion is null. Normally, the **`Fallback`** property for a given point defines the meaning of an Enum null as zero (0). If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null default value for the point. If this modified component links to another StatusEnum property, the framework ignores the null value. But, if connected to a StatusEnumToInt component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **`outValueOnNull`** property to this component. Configure it to return a reasonable value when the **`In`** value is null.
Out Value On Null	number	Defines an **`In`** value for the conversion object when the input **`Out`** value has a null status.

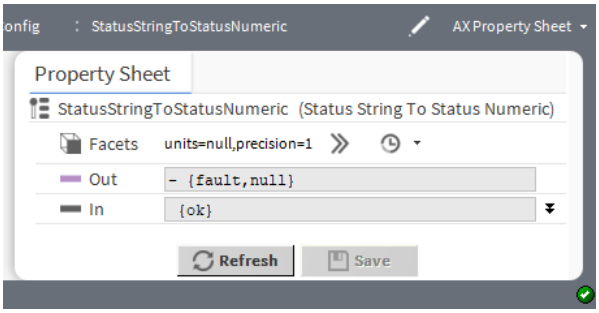
kitControl-StatusStringToStatusNumeric

This component converts a StatusString value to an StatusNumeric value.

StatusStringToStatusNumeric is available in the **Conversion** folder of the **`kitControl`** palette.

The input string must contain only numeral characters, and optionally one period (.) for decimal notation. Valid input strings examples are `145678` or `34.81`. Leading and/or trailing space characters are allowed and ignored. Any other input string characters (for example, alpha, punctuation) result in a fault status.

Figure 40 StatusStringToStatusNumeric properties



Property	Value	Description
Facets	Config Facets window (defaults to null for units and one decimal place of precision)	Selects unit and configures how the value displays: `units` define the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **`In`** property value.
Out	read-only number	Displays the converted target status numeric value.
In	number and null definition (defaults to 0)	Displays the source status text.

Property	Value	Description
		When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
On Null In Value	drop-down list (defaults to Use In Value)	Configures how to handle null values. `Use In Value` outputs the **Out** default value when the result of the conversion is null. Normally, the **Fallback** property for a given point defines the meaning of null as 0.0. If, during configuration, the null status check box is unchecked and another value entered and saved, the changed value becomes the new null value for the point. If this modified component links to another StatusString property, the framework ignores the null value. But, if connected to a StatusStringToStatusNumeric component, the result of a null input value defaults to the changed value, which may not be what you intended. `Specify Out Value` adds the **Out Value On Null** property to this component. Configure it to return a reasonable value when the **Out** value is null.
Out Value On Null	number	Defines an **In** value for the conversion object when its **Out** value has a null status.

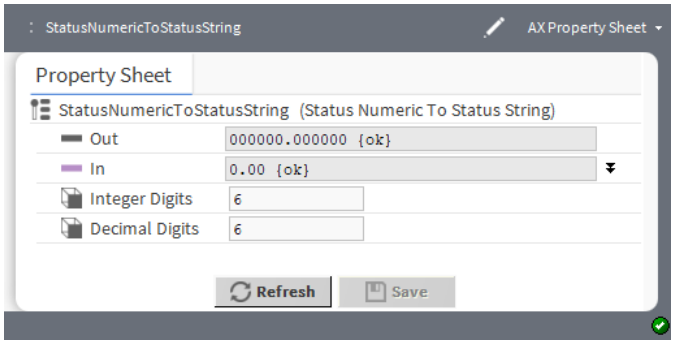
kitControl-StatusNumericToStatusString

This component converts a StatusNumeric value to a StatusString value.

StatusNumericToStatusString is available in the **Conversion** folder of the **kitControl** palette.

Object properties specify the number of integer (whole number) digits and decimal digits to use in the output string, with a default value of six (6) each. Unused digits are padded with a zero. For example, with an input numeric value of 72.8, the default output string would be: `000072.800000`.

Figure 41 StatusNumericToStatusString properties



Property	Value	Description
Out	read-only number	Displays the converted target status text.
In	number and null definition (defaults to 0)	Displays the source status numeric value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

Property	Value	Description
Integer Digits	number (defaults to 6)	Configures the whole number to the left of the **In** value's decimal point.
Decimal Digits	number (defaults to 6)	Configures the fraction to the right of the **In** value's decimal point.

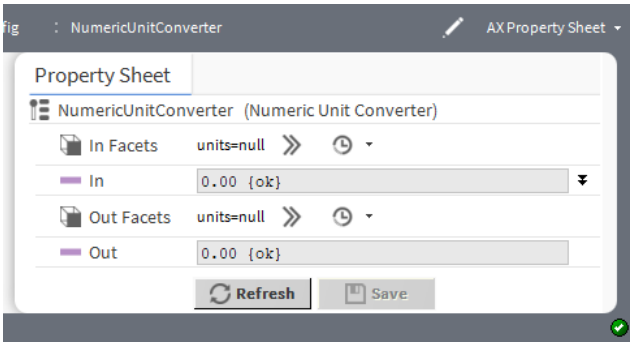
kitControl-NumericUnitConverter

This component converts a StatusNumeric value from a definable **In Facets** to a different definable **Out Facets**.

NumericUnitConverter is available in the **Conversion** folder of the **kitControl** palette.

This component is unique in that it does not change the data type (both input and output are StatusNumeric). It changes the actual numeric value. To produce a valid numeric output, both configured facets must conform to the same category (such as temperature, power, and so forth). Otherwise, the NumericUnitConverter reports a fault status.

Figure 42 NumberUnitConverter properties



Property	Value	Description
In Facets	Config Facets window	Configures the unit of measure for the **In** value.
In	number and null definition (defaults to 0)	Displays the source value based on the source unit of measure. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Out Facets	Config Facets window	Configures the unit of measure for the **Out** value.
Out	read-only number	Displays the converted out value based on its unit of measure.

Chapter 6 Energy components

Topics covered in this chapter

- ♦ kitControl-DegreeDays
- ♦ kitControl-ElectricalDemandLimit
- ♦ kitControl-NightPurge
- ♦ kitControl-OptimizedStartStop
- ♦ kitControl-OutsideAirOptimization
- ♦ kitControl-Psychrometric
- ♦ kitControl-SetpointLoadShed
- ♦ kitControl-SetpointOffset
- ♦ kitControl-ShedControl
- ♦ kitControl-SlidingWindowDemandCalc

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. These components include a degree-days calculation object as well as various objects used for electrical demand limiting. Additional energy-saving functions are also represented as components.

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

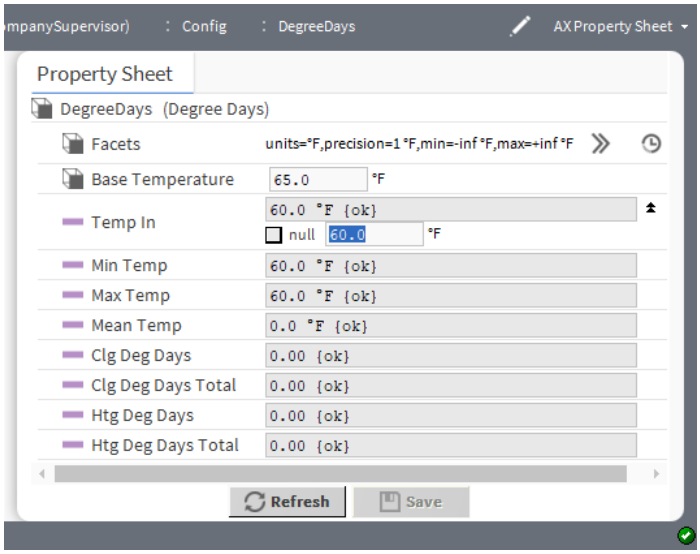
kitControl-DegreeDays

This component provides degree day calculations, based upon temperature received at the Temp In slot and values of various other properties.

DegreeDays is available in the **Energy** folder of the **`kitControl`** palette.

Degree Days is a unit of measure that may be expressed as either Heating Degree Days (HDD) or Cooling Degree Days (CDD). You calculate Degree Days by taking the difference between the average temperature during a given time period (month, season, year) and a reference point, usually 65 degrees Fahrenheit. Both cooling and heating degree day values are available, including totalized values. A Reset Totals action is available to clear (zero) totalized values.

Figure 43 DegreeDays property sheet



The screen capture is an example of the DegreeDays property sheet.

Property	Value	Description
Facets	Config Facets window (defaults to Fahrenheit, a single decimal place, negative and positive infinity respectively)	Selects units and configures how the value displays: `units` defaults to temperature. `precision` defines number of decimal places in temperature. `min` defines the lowest temperature allowed. `max` defines the highest temperature allowed. This configuration applies to the **`Temp In`**, **`Min Temp`**, **`Max Temp`**, and **`Mean Temp`** properties.
Base Temperature	degrees	Specifies the base temperature used in the degree-day calculation.
Temp In	number of degrees and null definition (defaults to 0)	Defines the input for the outside air temperature used in the degree-day calculation. If this input is not valid temperature, the framework performs no calculations. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Min Temp	read-only degrees	Displays the minimum temperature recorded for the current day. The framework tests and sets this value on each calculation.
Max Temp	read-only degrees	Displays the maximum temperature recorded for the current day. The framework tests and sets this value on each calculation.
Mean Temp	read-only degrees	Displays the mean temperature recorded for the previous day. The framework calculates this value when the day changes. The calculation is: Temp = (Max Temp + Min Temp) / 2.0

Property	Value	Description

Clg Deg Days	read-only degrees	Displays the cooling degree-day calculated for the previous day. The framework calculates this value when the day changes using this formula:
		`If (Mean Temp - Base Temperature) > 0`
		`Clg Deg Days = Mean Temp - Base Temperature`
		`else`
		`Clg Deg Days = 0.0`

Clg Deg Days Total	read-only degrees	Displays the totalized cooling degree-days since the user invoked the last `Reset Totals` action. The framework calculates this value when **`Clg Deg Days`** changes.

Htg Deg Days	read-only degrees	Displays the heating degree-day calculated for the previous day. The framework calculates this value when the day changes using this formula:
		`If (Mean Temp - Base Temperature) < 0`
		`Htg Deg Days = Base Temperature - Mean Temp`
		`else`
		`Htg Deg Days = 0.0`

Htg Deg Days Total	read-only degrees	Displays the totalized heating degree-days sincethe user invoked the last `Reset Totals` action. The framework calculates this value when Htg Deg Days changes.

kitControl-ElectricalDemandLimit

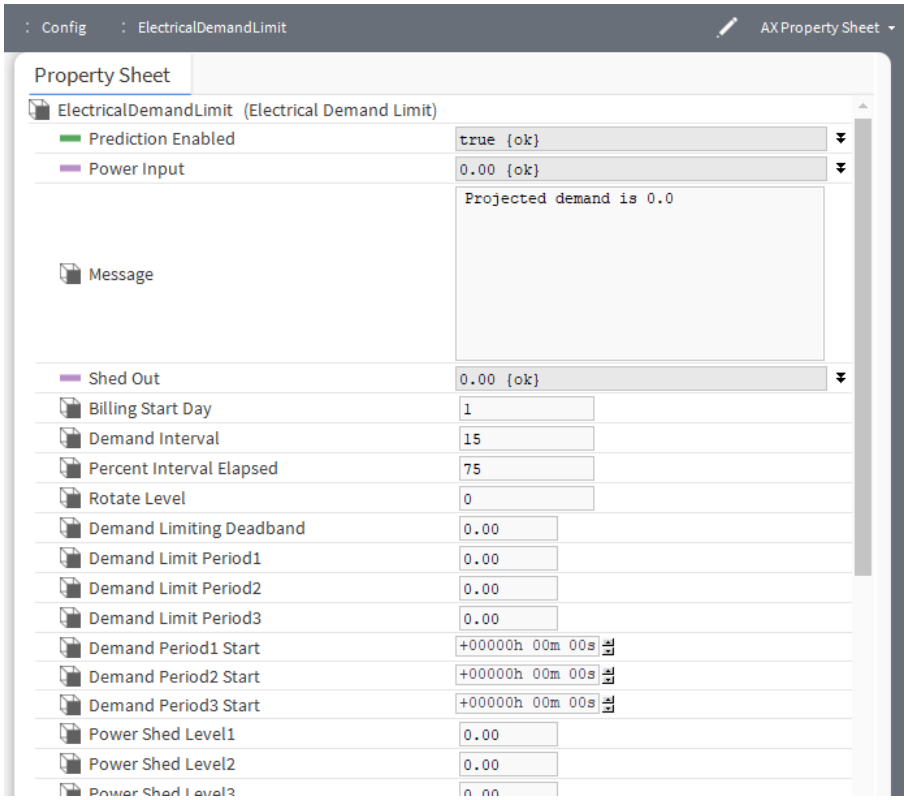
This component provides load shedding calculations based upon projected electrical demand averaging. It uses a sliding window interval, which can be configured.

ElectricalDemandLimit is available in the **Energy** folder of the **`kitControl`** palette along with related objects SetpointOffset and ShedControl.

Electrical Demand Limiting (EDL) is an energy management tool for leveling out fluctuations that may occur in daily energy demand levels. Energy providers often set billing rates based on periodic maximum demand levels. It is possible that a single day (an anomaly) could dramatically increase a monthly billing rate. Reducing peak demand levels can significantly lower energy costs even if the total energy consumption does not change. To avoid costly spikes in demand level, this component monitors and controls building and enterprise energy usage.

To use the ElectricalDemandLimit component, as a minimum, you need to do the following:

Figure 44 ElectricalDemandLimit configuration properties



Property	Value	Description
Prediction Enabled	`true` (default) or `false` and null definition	Turns this component on and off. `true` enables this component. This value must be set to `true` for the component to work. `false` disables this component. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Power Input	kW rate and null definition (defaults to 0)	Defines the power demand (kW) rate. The component uses this value to average the demand rate over every minute for comparison with the **`Projected Demand Average`**. This property should represent the total actual demand rate — the total of all meters monitored by this component on the energy network. When the component invokes a shed or restoration, this value is expected to change in relation to the estimated values configured for the **`Power Shed Level`** properties. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Message	read-only text	Displays information that relates to the status of the shed, restoration, or projected demand values, and may indicate the status of the EDL component, itself.

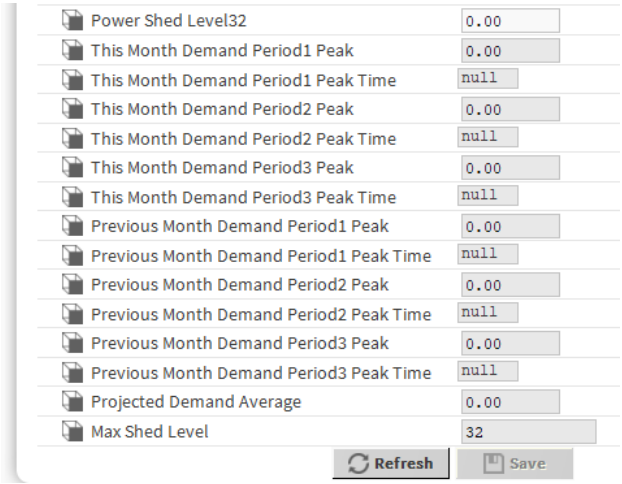
Property	Value	Description
Shed Out	number and null definition (defaults to 0)	Displays a value that indicates the number of levels to be shed. For example, a **Shed Out** value of 3 indicates that a Power Shed Level of 3 is being shed. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**In** value.
Billing Start Day	date	Specifies the first utility billing day of the month. This aligns system data with the actual energy company billing periods. Each month, on the day specified by this property, the current month's data move to the previous month, and the current month's data become this month.
Demand Interval	15 or 30 minutes (defaults to 15 minutes)	Configures the demand window used to calculate the average. You may change this value to 30 minutes. Any other value reverts to 15 minutes. The larger demand interval has more data points (one per minute) than the smaller interval. Depending on the value of the **Percent Interval Elapsed** property, the data points may be comprised of mostly sampled historical demand values, mostly values that are projected (based on current demand), or half and half.
Percent Interval Elapsed	percentage (defaults to 75)	Determines how much of the calculated demand is based on actual demand as opposed to how much is based on projected demand. A value of 50 calculates the **Projected Demand Average** using half of the actual sampled (or historical) values and half of the projected values. To calculate projected demand, the component uses current energy data at the current minute. In a 15 minute demand window, a value of 67 means that the current minute is 10. Larger numbers in this property increase the amount of historical data used and decrease the amount of data that are based on the current minute demand.
Rotate Level	a number from 1 to 32 (defaults to 0)	Specifies the maximum **Shed Level** to use. For example, a value of 3 limits load shedding to **Power Shed Level3**.
Demand Limiting Deadband	kW	Sets a deadband value the component uses to determine whether or not to invoke a restoration when activating restoration levels. The component does not use this value to invoke shed actions.
Demand Limit Period(n)	kW	These three values set demand limits that correspond to each of the three daily periods. When energy usage exceeds the **Demand Limit** value for a period, load shedding begins.

Property	Value	Description
Demand Period(n) Start	hours minutes seconds (default to 0)	These three values split up a 24-hour day into three time-periods used to assign a separate demand limit for each distinct time.
Power Shed Level (n)	a number to two decimal places	These levels set up to 32 demand limit values based on the loads that are controlled by the specific shed group. Each value represents the amount of demand to shed when the component invokes the associated shed level. The component may invoke one or more shed levels to lower the demand below the current **`Demand Period`** limit. In other words, the load-shed logic calculates how much demand is required to be reduced, and then uses these values to determine how many loads to shed. Once the component invokes a shed level is invoked, it evaluates the actual power drop at the next minute to determine the effect of the shed action. If the initial load shed does not actually reduce the demand below the limit, the component invokes the next shed level (if any). To limit the number of shed levels the component may invoke, configure the **`Rotate Level`** property.

ElectricalDemandLimit data – kitControl

You access these data by scrolling down the ElectricalDemandLimit property sheet.

Figure 45 ElectricalDemandLimit data



Data item	Description
This Month Demand Period(n) Peaks	Displays the value of the highest demand (minute) so far in the current month.
This Month Demand Period(n) Peak Times	Displays the time and date that the current month's peak demand occurred.
Previous Month Demand Period(n) Peaks	Displays the value of the highest demand (minute) that occurred in the previous month.
Previous Month Demand Period(n) Peak Times	Displays the time and date that the previous month's peak demand occurred.

Data item	Description
Projected Demand Average	Totals the value of the **Power Input** every time this value changes, and once a minute calculates **Projected Demand Average** and **Shed Level**. The component averages a combination of projected and historical samplings over the specified interval. To influence the value of the **Projected Demand Average**, configure the **Demand Interval** and the **Percent Interval Elapsed** properties.
Max Shed Level	Displays the maximum shed level used in the current month.

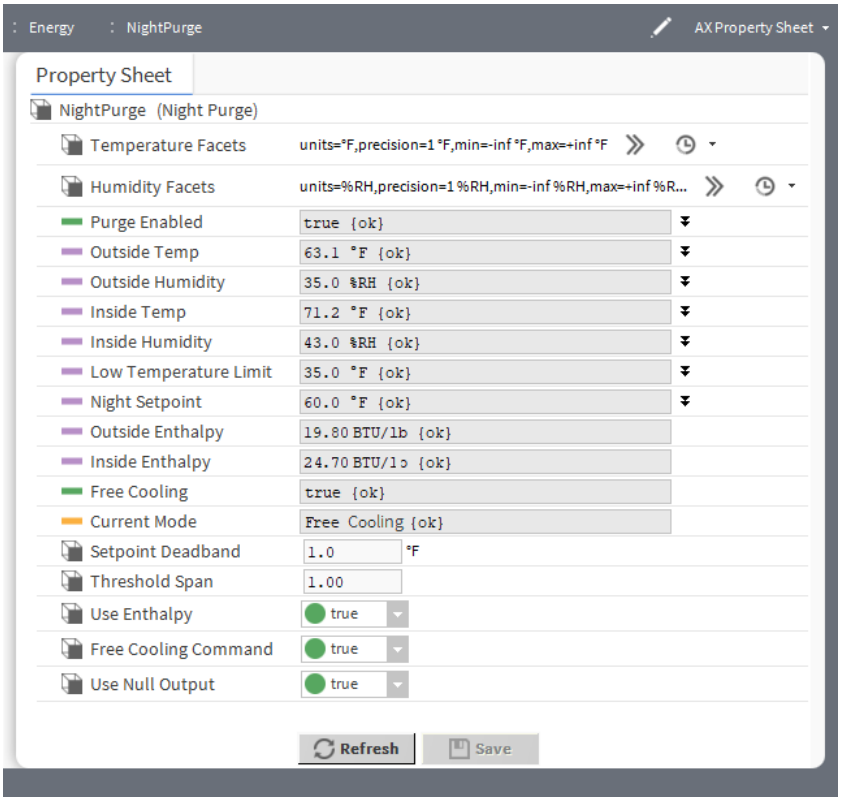
kitControl-NightPurge

This component uses the two sets of temperature and humidity inputs to find the air supply with the least amount of heat when the **Purge Enabled** input is `true`.

NightPurge is available in **Energy** folder of the **kitControl** palette.

For inside and outside comparisons, you can select either temperature or enthalpy comparisons. There is also a low temperature check to protect against freezing.

Figure 46 NightPurge properties



Property	Value	Description
Temperature Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to temperature. `precision` defines the number of decimal places of temperature. `min` defines the lowest temperature allowed. `max` defines the highest temperature allowed. This configuration applies to the **`Outside Temp`**, **`Inside Temp`**, and **`Low Temperature Limit`** property values.
Humidity Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to percentage of relative humidity (%RH) . `precision` defines the number of decimal places of humidity percentage. `min` defines the lowest humidity allowed. `max` defines the highest humidity allowed. This configuration applies to the **`Outside Humidity`** and **`Inside Humidity`** property values.
Purge Enabled	StatusBoolean `true` or `false` (default) and null definition	`true` enables the night purge operation. `false` sets the **`Free Cooling`** output to the opposite of the **`Free Cooling Command`** (or `null`, if **`Use Null Output`** is set to `true`), and the **`Current Mode`** slot value is `Disabled`. Often this slot is linked to a Not object sourced from a BooleanSchedule output. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Outside Temp	number of degrees of temperature (defaults to 0.0F) and null definition	Defines the current outside air temperature. This input must be valid for this object to function. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Outside Humidity	percentage of relative humidity (defaults to 0.0 %RH) and null definition	Defines the current outside air humidity. This input must be valid for this object to function. When null is checked, the corresponding value defines the default when a calculation returns null.
Inside Temp	number of degrees of temperature (defaults to 0.0F) and null definition	Defines the current inside air temperature. This input must be valid for this object to function. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Inside Humidity	percentage of relative humidity (defaults to 0.0 %RH) and null definition	Defines the current inside air humidity. This input must be valid for this object to function.

Property	Value	Description
		When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Low Temperature Limit	number of degrees of temperature (defaults to 0.0F) and null definition	Provides freeze protection. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Night Setpoint	number of degrees of temperatur (defaults to 0.0F) and null definition	Defines the night temperature setpoint, at or below which free cooling is not applied. Instead, the **Current Mode** is set to `Satisfied`. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Outside Enthalpy	read-only energy measured in BTUs per pound to two decimal places (defaults to 0.00 BTU/lb)	Displays the calculated outside air enthalpy.
Inside Enthalpy	read-only energy measured in BTUs per pound to two decimal places (defaults to 0.00 BTU/lb)	Displays the calculated inside air enthalpy.
Free Cooling [output]	read-only Status-Boolean `true` or `false` (default)	Displays the value of the **Free Cooling Command** when free cooling should be used. Otherwise, it displays the opposite state, or `null` (if **Use Null Output** is set to `true`). The component sets the **Free Cooling** output to `false` if `outside >= inside` or `true` if `outside = Night Setpoint`.
Current Mode	read-only value (defaults to `Disabled`	Indicates which of the following modes this object is currently in: `Disabled` indicates that **Purge Enabled** is `false`. `Free Cooling` `No Free Cooling` indicates that free cooling is not available. `Low temperature` indicates that Outside Temp is below the Low Temperature Limit and free cooling is disabled. `Input error` indicates that a temperature or humidity is invalid (down, fault, etc.) and free cooling is disabled. `Satisfied` indicates that inside temperature is below the **Night Setpoint** and free cooling is disabled.
Setpoint Deadband	degrees of temperature to a single decimal place (defaults to 1.0F)	Applies temperature **Setpoint Deadband** when inside temperature falls below **Night Setpoint** before free cooling can be enabled.

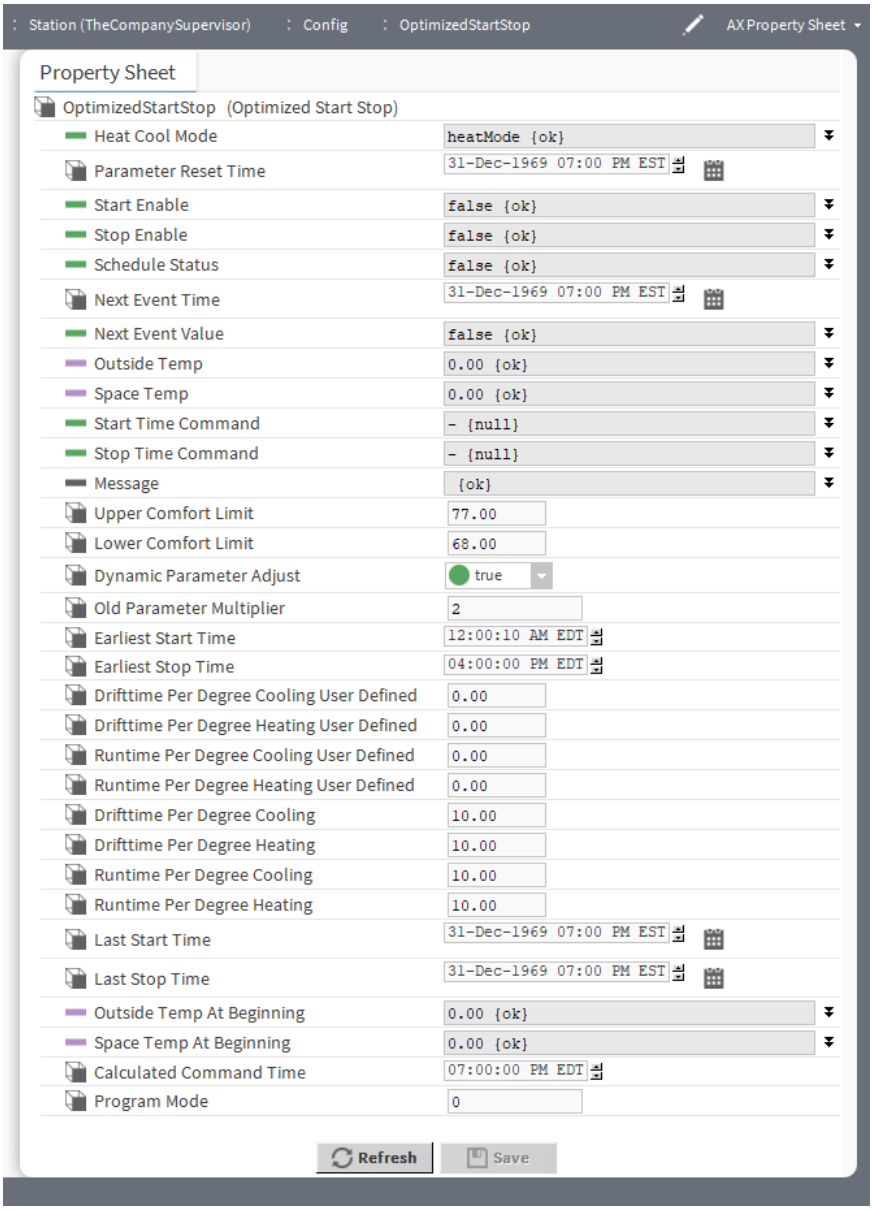
Property	Value	Description
Threshold Span	number to two decimal places (defaults to 1.0)	Defines a value. The difference between the **Inside Enthalpy** and the **Outside Enthalpy** must be greater than this value before the component enables free cooling.
Use Enthalpy	StatusBoolean `true` (default) or `false`	`true` enables the use of enthalpy to determine if free cooling is available. `false` uses outside and inside temperature to decide.
Free Cooling Command	StatusBoolean `true` (default) or `false`	Configures **Free Cooling** output. `true` indicates that free cooling is available, therefore **Free Cooling** output should be set to `true`. `false` indicates that free cooling is not available, therefore **Free Cooling** output should be set to `false`.
Use Null Output	StatusBoolean `true` (default) or `false`	Configures **Free Cooling** output. `true` sets the null flag for **Free Cooling** output when free cooling is not available. `false` ignores the null flag for **Free Cooling** output.

kitControl-OptimizedStartStop

This component uses **Start Time Optimization** and **Stop Time Optimization** to save energy. It uses a space temperature input and area characteristics to calculate an optimal amount of lead-time before a scheduled event. It can analyze area temperature changes and adjust the optimization parameters based on the actual temperature change rates after an optimized start or stop.

OptimizedStartStop component is available in the **Energy** folder of the **kitControl** palette.

Figure 47 OptimizedStartStop properties



Property	Value	Description
Heat Cool Mode	drop-down list and null definition	Selects heating or cooling. The selected option applies only to optimized stop calculations. This means that optimized stop calculations are performed only for the selected mode. Optimized start calculations are performed for both heat and cool modes regardless of this property value. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Parameter Reset Time	date hour:minute (defaults to 7 PM EST (Eastern	Displays the time when any of the four runtime or driftime properties change to the user-defined values. This component copies the user defined drifttime and runtime property values

Property	Value	Description
	Standard Time); includes date picker	to the corresponding actual drifttime and runtime property values.
Start Enable	`true` and `false` (default) and null definition	Manually or automatically turns the optimized start function on (`true`) and off (`false`). When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Stop Enable	`true` and `false` (default) and null definition	Manually or automatically turns the optimized stop function on (`true`) and off (`false`). When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Schedule Status	`true` and `false` (default) and null definition	Monitors and displays the status of the schedule that is linked to this component. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Next Event Time	date and time (defaults to EST (Eastern Standard Time))	Links to a schedule for the time of the next scheduled event.
Next Event Value	`true` and `false` (default) and null definition	Links to a schedule and reflects the value of the action for the next scheduled event. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Outside Temp	number of degrees (defaults to 0.00) and null definition	Links to outside temperature and displays the value for information only. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Space Temp	number of degrees (defaults to 0.00) and null definition	Links to a space temperature output and displays the temperature of the area affected by equipment associated with this component. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Start Time Command	`true` and `false` (default) and null definition	Links to a control for invoking an equipment start command. For example, you can link it to a prioritized input of a Boolean writable, or directly to the equipment Start control. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

Property	Value	Description
Stop time Command	`true` and `false` (default) and null definition	Links to a control for invoking an equipment stop command. For example, you can link it to a prioritized input of a Boolean writable, or directly to the equipment Stop control. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Message	text and null definition	Provides information that indicates the results of the latest start or stop command, the status of an optimized start analysis, or other possible messages. For example, the following message indicates that an optimized stop has occurred: `Optimized stop for 14-Jun-07 5:18 PM EDT schedule time. Space temp is 75.0.` When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Upper Comfort Limit	degrees to two decimal places (defaults to 0.00) (defaults to 77.00 F, 25.00 C)	Defines the cooling mode target temperature.
Lower Comfort Limit	degrees to two decimal places (defaults to 0.00) (defaults to 68.00 F, 20.00 C)	Defines the heating mode target temperature.
Dynamic Parameter Adjust	`true` (default)or `false`	Controls if calculation the component programmatically adjusts parameters after an execution. After the component completes a start or stop control, if this property is set to `true`, the component evaluates the actual recovery rate (degrees/hour) and automatically adjusts the values of the runtime and drifttime properties so that they are influenced by actual drift time and run time.
Old Parameter Multiplier	single-digit number (defaults to 2)	Applies a weighting factor to the dynamic parameter adjustment calculation. This value affects how much weight to assign to the previous runtime property value when the component uses it in a dynamic parameter adjustment calculation. A larger value increases the weighting factor given to the previous runtime value, and a smaller value decreases this weighting factor.
Earliest Start Time	hours minutes seconds (defaults to 12 and 10 seconds AM EDT (Eastern Daylight Time))	Specifies a time, before which, the component may issue no optimized start command. If this value is set earlier than the **`Calculated Command Time`**, the component adjusts the **`Calculated Command Time`** to equal this time.
Earliest Stop Time	hours minutes seconds (defaults to 4 PM EDT (Eastern Daylight Time))	Specifies a time, before which, the component may issue no stop command. If this value is set earlier than the **`Calculated Command Time`**, the component adjusts the **`Calculated Command Time`** to equal this time.

Property	Value	Description
Drifttime Per Degree cooling User Defined	single-digit number to two decimal places (defaults to 0.00)	Sets a default value for calculating the rate of drift in cooling mode. The component copies this value to the **Drifttime Per Degree Cooling** property.
DriftTime Per Degree Heating User Defined	single-digit number to two decimal places (defaults to 0.00)	Sets a default value for calculating the rate of drift in heating mode. The component copies this value to the **Drifttime Per Degree Heating** property.
Runtime Per Degree cooling User Defined	single-digit number to two decimal places (defaults to 0.00)	Sets a default value for calculating the runtime value in cooling mode. The component copies this value to the **Runtime Per Degree Cooling** property.
Runtime Per Degree Heating User Defined	single-digit number to two decimal places (defaults to 0.00)	Sets a default value for calculating the runtime value in heating mode. The component copies this value to the **Runtime Per Degree Heating** property.
Drifttime Per Degree Cooling	two-digit number to two decimal places (defaults to 10.00)	Displays the actual value used to calculate an optimized stop time when the equipment is in cooling mode. The component automatically adjusts this value **Dynamic Parameter Adjust** value is set to `true`.
Drifttime Per Degree Heating	two-digit number to two decimal places (defaults to 10.00)	Displays the actual value used to calculate an optimized stop time when the equipment is in heating mode. The component automatically adjusts this value if **Dynamic Parameter Adjust** value is set to `true`.
Runtime Per Degree Cooling	two-digit number to two decimal places (defaults to 10.00)	Displays the actual value used to calculate an optimized start time when the equipment is in cooling mode. The component automatically adjusts this value if **Dynamic Parameter Adjust** value is set to `true`.
Runtime Per Degree heating	two-digit number to two decimal places (defaults to 10.00)	Displays the actual value used to calculate an optimized start time when the equipment is in heating mode. The component automatically adjusts this value if **Dynamic Parameter Adjust** value is set to `true`.
Last Start Time	date, hours:minutes (defaults to 7 PM EST (Eastern Standard Time)	Displays the last start time used to calculate an optimized start time. Since the component allows only one optimized start per day, this value does not display start times (restarts) that occur subsequent to the initial start time for the day.
Last Stop Time	date, hours:minutes (defaults to 7 PM EST (Eastern Standard Time))	Displays the last stop time used to calculate an optimized stop time. Since the component allows multiple optimized stops per day, this value changes to reflect the latest optimized stop time.
Outside Temp At Beginning	number of degrees (defaults to 0.00) and null definition	Displays the outside air temperature at the time of the last start or stop command. This is the temperature used in the dynamic parameter adjustment calculations. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

Property	Value	Description
Space Temp At Beginning	number of degrees (defaults to 0.00) and null definition	Displays the space temperature at the time of the last start or stop command. This is the temperature used in the dynamic parameter adjustment calculations. When null is checked, the corresponding value defines the default when a calculation returns null.
Calculated Command Time	hours:minutes (defaults to 7 PM EST (Eastern Standard Time))	Shows the calculated time for the next command. This could be a start or a stop command.
Program Mode	read-only single-digit number (defaults to zero (0))	Displays one of five current heating or cooling states for an optimized start or stop. These states serve primarily in logic control and may also inform the system engineer. `0` (no calculation) indicates that no calculation is in progress. `1` (start calculation) indicates that the optimized start calculation is ongoing, but no optimized start or stop is yet in progress. `2` (start in process) indicates that the component initiated an optimized start. `3` (stop calculation) indicates that an optimized stop calculation is ongoing, but no optimized start or stop is yet in progress. `4` (stop in process) indicates that the component initiated an optimized stop.

kitControl-OutsideAirOptimization

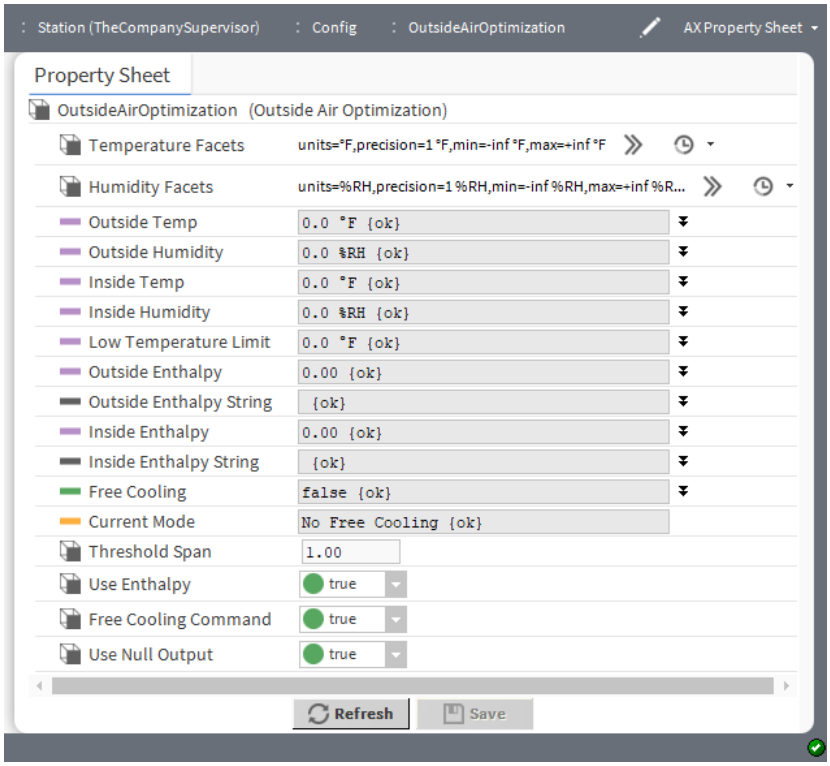
This component supports applications that need to allow for enthalpy based free cooling. This object is typically used during occupancy periods.

OutsideAirOptimization is available in the **Energy** folder **`kitControl`** palette.

The freeCooling output is set to `false` if `outside` >= `inside` and set to true if `outside` <= `inside` – `(abs) thresholdSpan`.

There is also a low temperature check to protect against freezing.

Figure 48 OutsideAirOptimization properties



Property	Value	Description
Temperature Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to temperature. `precision` defines the number of decimal places of temperature. `min` defines the lowest temperature allowed. `max` defines the highest temperature allowed. This configuration applies to **`Outside Temp`**, **`Inside Temp`**, and **`Low Temperature Limit`** properties.
Humidity Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to percentage of relative humidity (%RH). `precision` defines the number of decimal places of humidity percentage. `min` defines the lowest humidity allowed. `max` defines the highest humidity allowed. This configuration applies to the **`Outside Humidity`** and **`Inside Humidity`** property values.
Outside Temp	number of degrees of temperature (defaults to 0.0 degrees F) and null definition	Defines the current outside air temperature. This input must be valid for this object to function.

Property	Value	Description
		When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Outside Humidity	percentage of humidity (defaults to 0.0 RH percent) and null definition	Defines the current outside air humidity. This input must be valid for this object to function. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Inside Temp	number of degrees of temperature (defaults to 0.0 degrees F) and null definition	Defines the current inside air temperature. This input must be valid for this object to functio
Inside Humidity	percentage of humidity (defaults to 0.0 RH percent) and null definition	Defines the current inside air humidity. This input must be valid for this object to function.
Low Temperature Limit	number of degrees of temperature (defaults to 0.0 degrees F) and null definition	Provides freeze protection. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Outside Enthalpy	number (defaults to 0.00) and null definition	Displays the calculated outside air enthalpy. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Outside Enthalpy String	text (defaults to blank) and null definition	Provides the outside enthalpy value as a string or possible status/error message. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Inside Enthalpy	number (defaults to 0.00) and null definition	Displays the calculated inside air enthalpy. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Inside Enthalpy String	text (defaults to blank) and null definnition	Provides the inside enthalpy value as a string or possible status/error message. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Free Cooling	`true` or `false` (default) and null definition	Sets the value of the **Free Cooling Command** when the component determines that free cooling should be used. Otherwise, the value is null. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

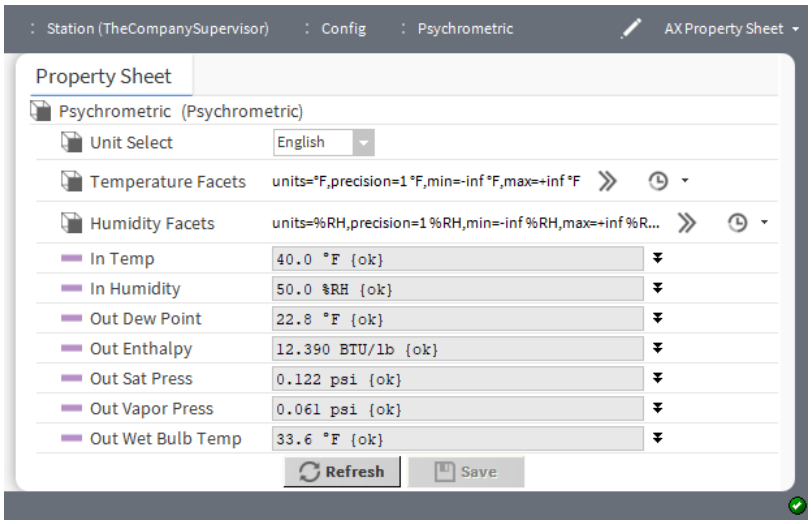
Property	Value	Description
Current Mode	text (defaults to `No Free Cooling`)	Indicates what mode this object is currently in: Input out of range Free Cooling No Free Cooling Low temperature Input error
Threshold Span	number to two decimal places (defaults to 1.00)	Configures the difference between the inside enthalpy and the outside enthalpy. The actual difference must be greater than this value before the system enables free cooling.
Use Enthalpy	`true` (default) or `false`	Enables (`true`) and disables (`false`) the use of enthalpy for determining if free cooling is available. Otherwise, the component uses outside and inside temperature to decide.
Free Cooling Command	`true` (default) or `false`	Defines the Boolean value to be set in the **`Free Cooling`** property if free cooling is available.
Use Null Output	`true` (default) or `false`	`true` sets the null flag on **`Free Cooling`** output when free cooling is not available.

kitControl-Psychrometric

This component You can use it to support applications that need to calculate the properties of moist air using given temperature and humidity inputs.

Psychrometric is available in the **Energy** folder of the **`kitControl`** palette.

Figure 49 Psychrometric properties



Property	Value	Description
Unit Select	drop-down list (defaults to `English`)	Selects units of measure.
Temperature Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to temperature. `precision` defines the number of decimal places of temperature. `min` defines the lowest temperature allowed. `max` defines the highest temperature allowed. This configuration applies to **`Outside Temp`**, This configuration applies to the **`In Temp`** property value.
Humidity Facets	Config Facets window	Selects units and configures how the value displays: `units` defaults to percentage of relative humidity (%RH). `precision` defines the number of decimal places of humidity percentage. `min` defines the lowest humidity allowed. `max` defines the highest humidity allowed. This configuration applies to **`In Humidity`** property value.
In Temp	number of degrees of temperature (defaults to 0.0 degrees F) and null definition	Defines the input temperature. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
In Humidity	percentage of humidity (defaults to 0.0 percent RH) and null definition	Defines the input humidity. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Out Dew Point	nan (not a number) or degrees (defaults to nan) and null definition	Reports the calculated dew point temperature. This value requires valid **`In Temp`** and **`In Humidity`** values. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Out Enthalpy	nan (not a number) or BTU/lb (defaults to nan) and null definition	Reports calculated enthalpy. This value requires valid **`In Temp`** and **`In Humidity`** values. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.
Out Sat Press	number of psi (pounds per square inch) (defaults to 0.019) and null definition	Reports calculated saturated pressure. This value requires a valid **`In Temp`**. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

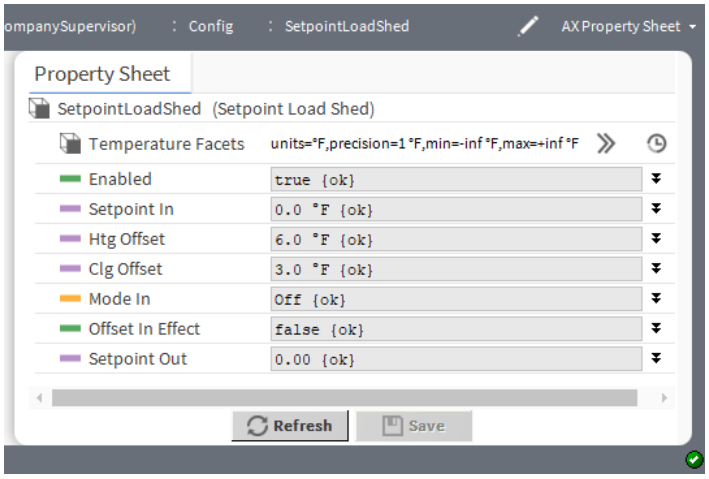
Property	Value	Description
Out Vapor Press	nan psi and null definition	Reports calculated vapor pressure. This value requires valid **`In Temp`** and **`In Humidity`** values. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Out Wet Bulb Temp	nan or degrees (defaults to nan) and null definition	Reports calculated wet bulb temperature. This value requires valid **`In Temp`** and **`In Humidity`** values. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

kitControl-SetpointLoadShed

This component provides a convenient way to implement load shedding strategies. It causes a specified set-point to be raised or lowered by a specific amount in response to an input link, that enables the SetpointLoadShed component. Generally, this component solves the problem often found in temperature control sequences where shutting down an output directly may be complicated by other control dependencies. By changing the setpoint based on a shed request, the control sequence directs the output when to shut down and maintains the interdependencies.

SetpointLoadShed is available in the **Energy** folder of the **`kitControl`** palette.

Figure 50 SetpointLoadShed property sheet



Property	Value	Description
Enabled	`true` (default) or `false` and null definition	Turns changing the setpoint on and off. `true` adjusts the setpoint by the offset value. `false` ignores the setpoint. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.
Setpoint In	number of degrees of temperature	Displays the normal setpoint value to be adjusted by this object.

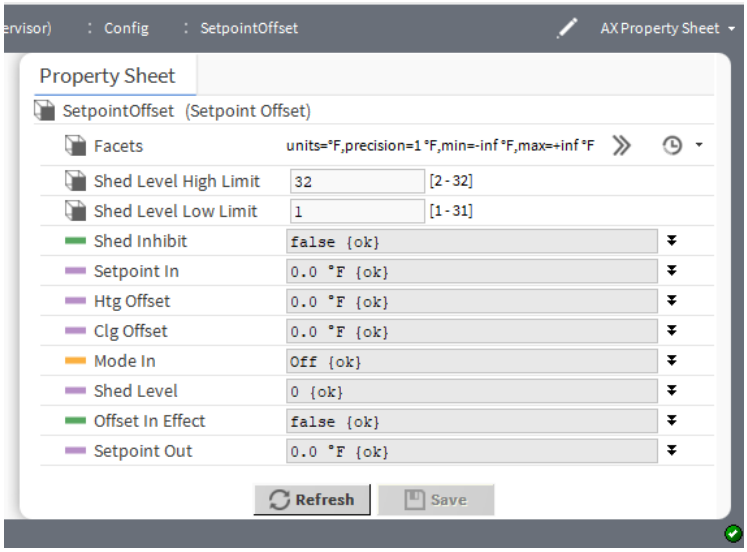
Property	Value	Description
	(defaults to 0.0 F) and null definition	When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Htg Offset	signed (+ or -) number of degrees of temperature (defaults to 6.0 F) and null definition	Defines the amount by which to adjust the setpoint when in heating mode. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Clg Offset	signed (+ or -) number of degrees of temperature (defaults to 3.0 F) and null definition	Defines the amount by which to adjust the setpoint when in cooling mode. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Mode In	StatusEnum (drop-down list) with three options and null definition	Selects the current need: Off, Heat, Cool. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Offset in Effect	`true` or `false` (default) and null definition	Indicates if the **Setpoint Out** property value has been adjusted. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Setpoint Out	number of degrees of temperature and null definition	Displays the adjusted setpoint value, if active, otherwise it passes through to the original setpoint. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

kitControl-SetpointOffset

This component provides setpoint control for electrical demand limiting applications. You use it with the ElectricalDemandLimit and ShedControl components.

SetpointOffset is available in the **Energy** folder of the **kitControl** palette

Figure 51 SetpointOffset properties



Property	Value	Description
Facets	Config Facets window (defaults to Fahrenheit temperature for units, a single decimal place, negative infinity for minimum and positive infinity for maximum)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. min and max refer to minimum and maximum temperatures.
Shed Level High Limit	number between 2 and 32 (defaults to 32) based on the facets	Defines the maximum offset.
Shed Level Low Limit	number between 1 and 31 (defaults to 1) based on the facets	Defines the level at which the offset begins.
Shed Inhibit	`true` or `false` (defaults to `false`)	Turns adjustment of the setpoint on and off: `true` disables the shedding ability. `false` causes the setpoint to be adjusted by the offset.
Setpoint In	number of degrees (defaults to zero (0.0))	Displays the source setpoint value linked in to this component.
Htg Offset	number of degrees (defaults to zero (0.0))	Defines the maximum setpoint (plus or minus) offset if heating is active and in heating mode.
Clg Offset	number of degrees (defaults to zero (0.0))	Defines minimum setpoint (plus or minus) offset if cooling is active and in cooling mode.

Property	Value	Description
Mode In	`Off` (default) and `On`	Selects the current need: Off, Heat, Cool.
Shed Level	number of degrees (defaults to zero (0))	Displays the shed level that is in effect.
Offset In Effect	`true` or `false` (default) and null definition	Indicates if the **`Setpoint Out`** property value has been adjusted. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Setpoint Out	number of degrees (defaults to zero (0))	Displays the adjusted setpoint if it is active. Otherwise, it pass through the original setpoint.

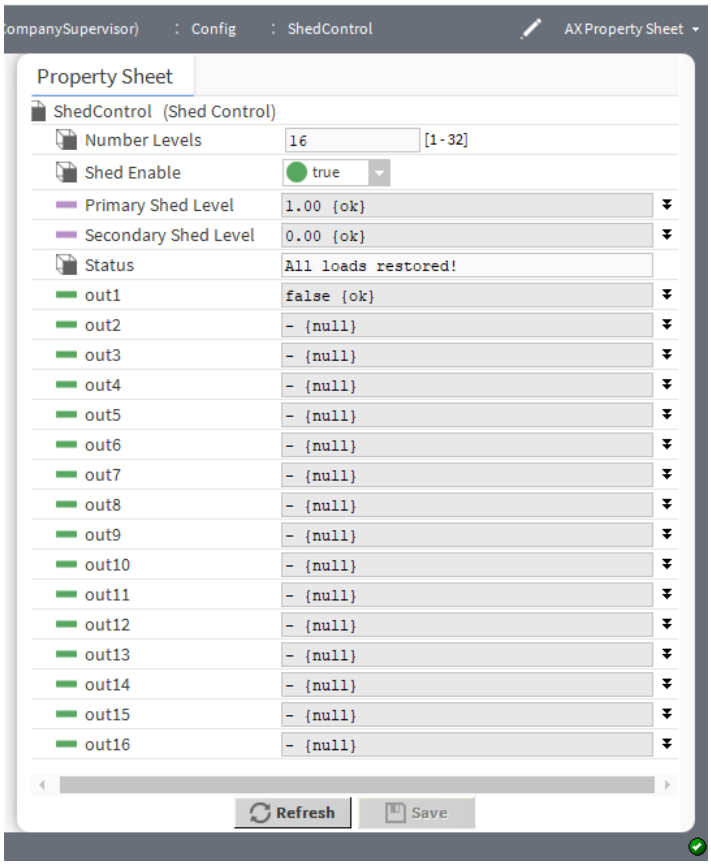
kitControl-ShedControl

This component receives inputs from a primary (network) EDL source and a local (secondary) EDL source (separate ElectricalDemandLimit objects) that specify the number of load levels that should be shed. The Secondary Shed Level backs up when the Primary Shed Level is not available

ShedControl is available in the **Energy** folder of the **`kitControl`** palette, along with related objects.

ShedControl has StatusBoolean outputs for up to 16 contiguous levels, as specified in the **`Number Levels`** property. A **`Status`** slot provides an output message to indicate this component's state in reference to the overall demand limiting control scheme.

Figure 52 ShedControlProperties



Property	Value	Description
Number Levels	number from 1–32 (defaults to 16)	Defines
Shed Enable	`true` (default) or `false`	Turns execution of this component on and off.
Primary Shed Level	number (defaults to 0.0) and null definition	Defines the input to link to from an EDL component. Typically, this would be a component on the network. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

Property	Value	Description
Secondary Shed Level	number (defaults to 0.0) and null definition	Defines an input to from a secondary (or backup) EDL component. Typically, this would be an EDL component with a locally available connection. The Secondary Shed level is used only if the Primary Shed Level property is not available. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**In** value.
Status (out1 through out16)	binary status values and null definitions	Reflect the current active Shed Level. For example, a Shed Level of 3 (as indicated by the Primary Shed Level, or Secondary Shed Level when Primary is not available) sets the first three out properties (out1, out2, out3) to false. This false value may be used to turn off power by linking to an appropriate control. When a restore changes the Shed Level to 2, the out3 property returns to null, relinquishing control to the next (out2) priority level. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**In** value.

kitControl-SlidingWindowDemandCalc

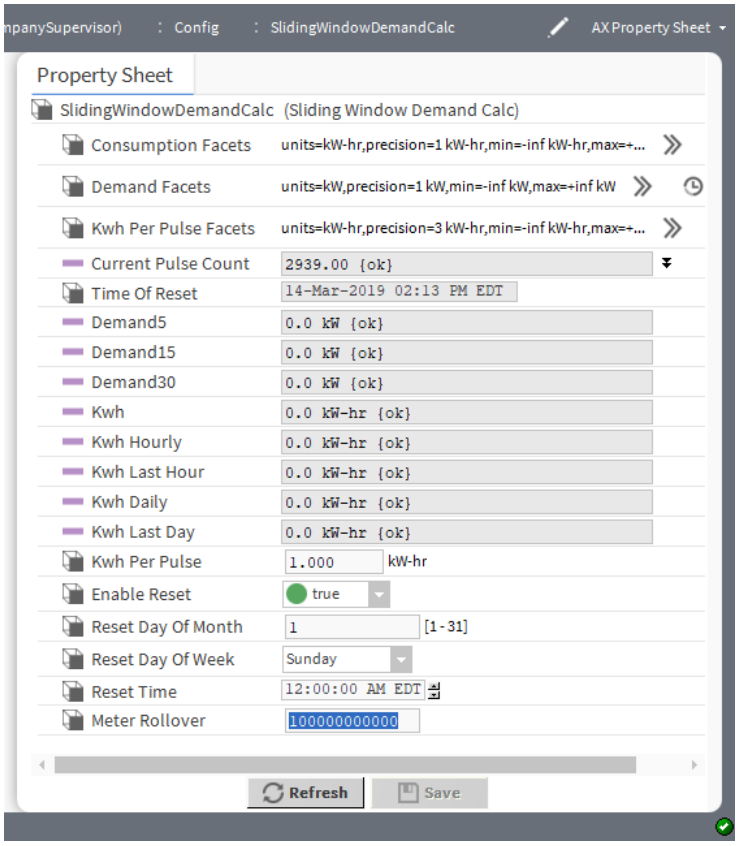
This component simulates a demand meter and calculates the sliding window demand for 5, 15, and 30-minute demand intervals based on an accumulative pulse counter input. It also calculates the total kWh since last reset and the hourly and daily kWh values.

SlidingWindowDemandCalc is available in the **Energy** folder of the **kitControl** palette.

Once calculated, a standard history extension setup to execute on-trigger only can log the hourly and daily kWh values. The SlidingWindowDemandCalc object fires the hourly and daily triggers to align the kWh data correctly to actual clock values. The pulse input into the object is assumed to be accumulative (not delta pulses) that roll over after reaching the 16 bit limit (65535).

You can configure this component to reset all the calculated accumulative values on a preset interval such as at “noon on the first Sunday, every month.”

Figure 53 SlidingWindowDemandCalc properties



Property	Value	Description
Consumption Facets	Config Facets window (defaults to kW-hr for units, one decimal point for precision, negative infinity for minimum value, and positive infinity for the maximum value)	Configures the consumption output property: `units` configures units of measure (default to energy and kilowatt hour (kW-hr)). `precision` defines the number of decimal places for the out values (defaults to 1). `min` defines the smallest allowable value of the output property (defaults to -infinity) `max` defines the largest allowable value of the output property (defaults to +infinity)
Demand Facets	Config Facets window (defaults to kW for units, one decimal point for precision, negative infinity for minimum, positive infinity for maximum)	Configures the demand output property: `units` configures units of measure (default to power and kilowatt hour (kW-hr). `precision` defines the number of decimal places for the out values (defaults to 1). `min` defines the smallest allowable value of the output property (defaults to -infinity) `max` defines the largest allowable value of the output property (defaults to +infinity)
Kwh Per Pulse Facets	Config Facets window (defaults to	Configures the kwh per pulse output property.

Property	Value	Description
	kW-hr for units, three decimal places for precision, negative infinity for minimum, and positive infinity for maximum)	`units` configures units of measure (default to energy and kilowatt hour (kW-hr)). `precision` defines the number of decimal places for the out values (defaults to 3). `min` defines the smallest allowable value of the output property (defaults to -infinity) `max` defines the largest allowable value of the output property (defaults to +infinity)
Current Pulse Count	number (defaults to 0.00) and null definition	Displays data from a link to a pulse counter input object indicating the running total of pulses. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Time of Reset	read-only	Displays the date and time of the last reset
Demand5	read-only	Displays the demand (kW) for a five minute window.
Demand15	read-only	Displays the demand (kW) for a fifteen minute window.
Demand30	read-only	Displays the demand for a 30 minute period.
Kwh	read-only	Displays the running kWh (consumption) value since the last reset.
Kwh Hourly	read-only	Displays the running value since the last hourly reset.
Kwh Last Hour	read-only	Displays the kWh (consumption) value for the last hour.
Kwh Daily	read-only	Displays the kWh (consumption) value since the last daily reset.
Kwh Last Day	read-only	Displays the kWh (consumption) value for the last day.
Kwh Per Pulse	number of kilowatts per hour (defaults to 1 kW-hr)	Sets the value per pulse, that is, how much energy each pulse represents. This value is usually noted on the meter or provided by the power company.
Enable Reset	`true` (default) or `false`	`true` configures recurring automatic resets to happen at a frequency based on the following properties. `false` disables automatic resets.
Reset Day of Month	number from 1 to 31	Sets the day of month when a recurring automatic reset (if enabled) occurs.
Reset Day of Week	drop-down list (defaults to Sunday)	Selects the day of the week when a recurring automatic reset (if enabled) occurs.
Reset Time	hours:minutes (defaults to 12 AM)	Sets the time of day for a recurring automatic reset (if enabled) to occur.
Meter Rollover	number	Specifies the maximum value the meter provides before it rolls over to zero (0). The default value is 65535, the data type is a long (up to a very large number, 9223372036854775807).

Chapter 7 HVAC components

Topics covered in this chapter

♦ kitControl-InterstartDelayControl
♦ kitcontrol-InterstartDelayMaster
♦ kitControl-LeadLagCycles
♦ kitControl-LeadLagRuntime
♦ kitControl-LoopPoint
♦ kitControl-RaiseLower
♦ kitControl-SequenceBinary
♦ kitControl-SequenceLinear
♦ kitControl-Tstat

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. These components provide various control functions used in commercial HVAC applications.

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

kitControl-InterstartDelayControl

This component delays the start of an HVAC component function.

InterstartDelayControl is available in the **HVAC** folder of the **`kitControl`** palette.

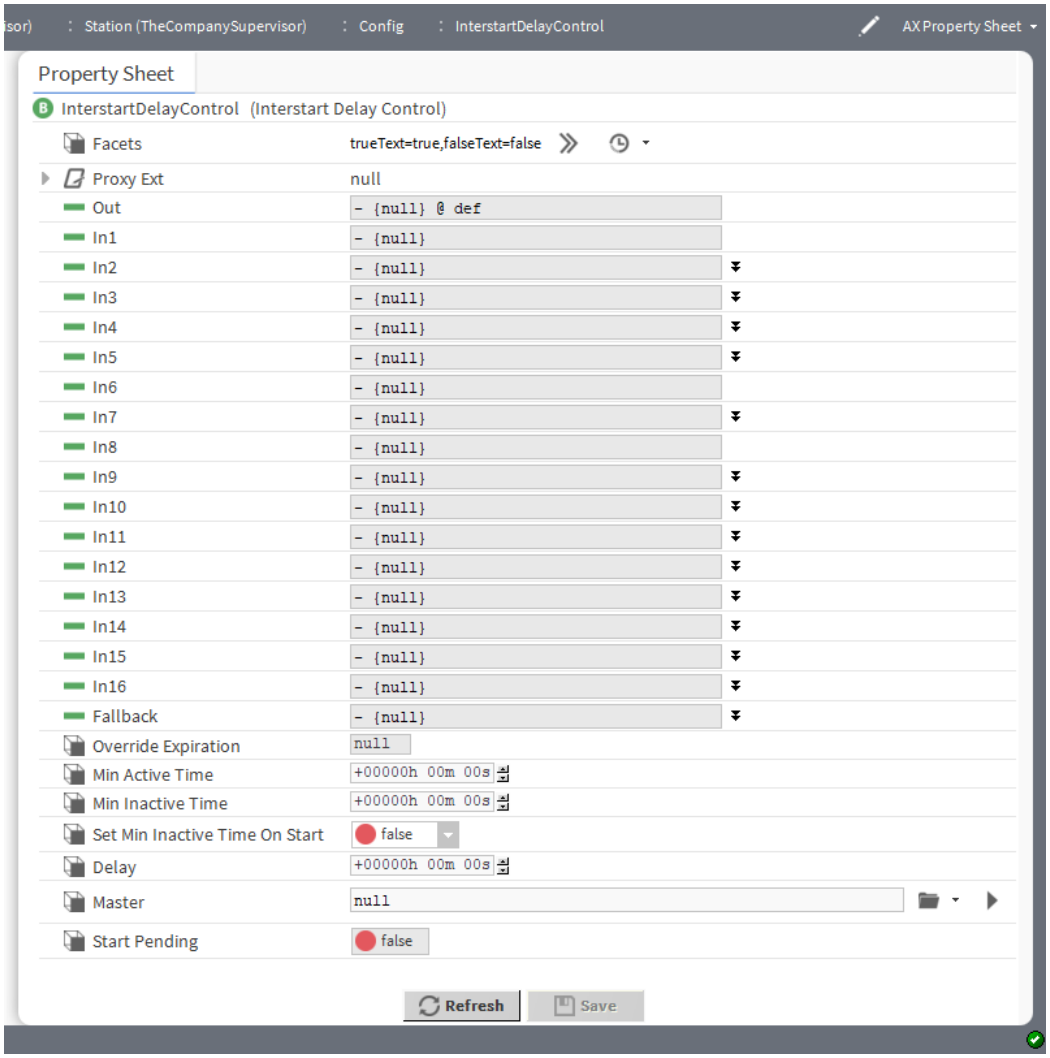
These objects are like BooleanWritables, each with three additional slots for use in interstart delay sequences, as follows:

- Delay — Specifies an amount of time to wait before the framework starts next object in the delay.
- Master — Specifies the InterstartDelayMaster component in the station that controls the delay sequence.
- Start Pending — a read-only Boolean status that indicates if a start is pending (`true`) or not (`false`).

Use this component with an InterstartDelayMaster, then link outputs of InterstartDelayControl objects, as needed, to control corresponding Boolean writable points (typically proxy points) for the final interstart control.

No other InterstateDelayControl object using the same delay master can start for delay time after this object starts. If delay is not defined, the framework uses the default delay on the master.

Figure 54 InterstartDelayControl properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns **`true`**. `falseText` configures the text to describe the state when the component returns **`false`**.
Out	read-only	Displays the result of the configuration.
In1 through In16	StatusBoolean `true` or `false` (default) and null definition	Link to up to 16 points, which provide StatusBoolean input. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Fallback	StatusBoolean `true` or `false` (default) and null definition	Sets the **`Out`** property's value to the value of this property when a point returns a null or invalid value at inputs **`In1-In16`**.

Property	Value	Description
Override Expiration	defaults to null	It defaults the value to null.
Min Active Time	hours minutes seconds	Provides for a minimum up time.
Min Inactive Time	hours minutes seconds	Provides for a minimum down time.
Set Min Inactive Time On Start	`true` or `false` (default)	Ensures the minimum inactive time when the station starts.
Delay	hours minutes seconds	Defines the amount of time to elapse before the component starts the next object in the sequence.
Master	file chooser (defaults to null)	Locates the InterstartDelayMaster component in the station.
Start Pending	read only Status-Boolean `true` or `false` (default)	Indicates if a start is pending. `true` indicates the start is pending. `false` indicates no start is pending.

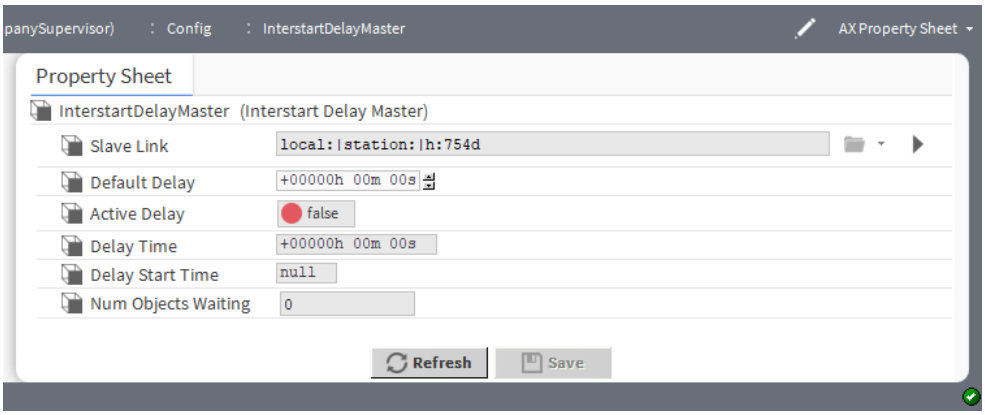
kitcontrol-InterstartDelayMaster

This component defines the master in an inter-start delay sequence. Use it in conjunction with one or more InterstartDelayControl objects.

InterstartDelayMaster is available in the **HVAC** folder of the **`kitControl`** palette.

An available action for this component is **`DelayTimerExpired`**.

Figure 55 InterstartDelayMaster properties



Property	Value	Description
Slave Link	ORD	Links to the slave InterstartDelayControl components.
Default Delay	hours minutes seconds (defaults to zero)	Defines the how long to wait before starting a request. The component uses this value only if **`Delay Time`** is not defined.
Active Delay	read-only `true` or `false` (default)	Displays if the delay is active. `true` indicates that the interstart delay is active

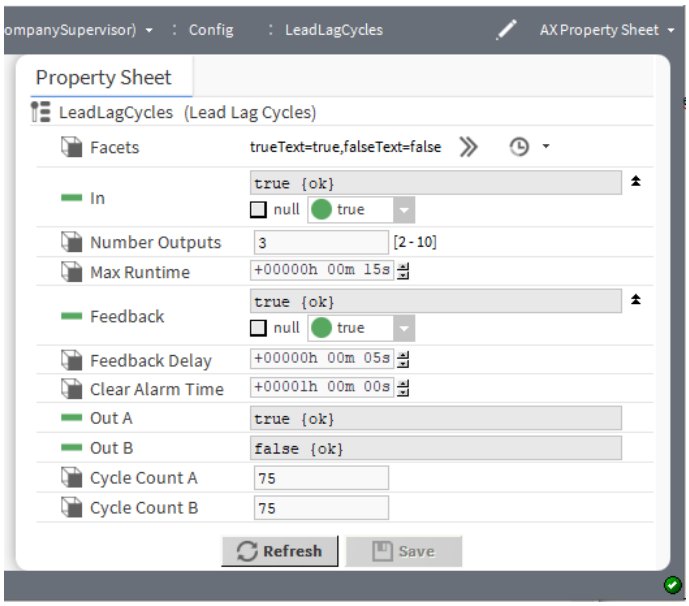
Property	Value	Description
		`false` indicates that the interstart delay is not active.
Delay Time	read-only hours minutes seconds (defaults to zero)	Displays the length of the current delay.
Delay Start Time	read-only clock time	Displays the time that the current delay started.
Num Objects Waiting	read-only number (defaults to zero (0))	Displays the number of objects currently waiting to start.

kitControl-LeadLagCycles

This component provides lead-lag control of from 2 to 10 BooleanWritables based upon their accumulated COS (change of state) counts. This object balances the number of change of states cycles of each of the devices. Only one of the controlled devices is active at a time based on cycle count

LeadLagCycles is available in the **HVAC** folder of the **`kitControl`** palette, along with a similar LeadLagRuntime object.

Figure 56 LeadLagCycles property sheet



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns **`true`**. `falseText` configures the text to describe the state when the component returns **`false`**.
In	status Boolean input: `true` or	Configures a control device to be on or off.

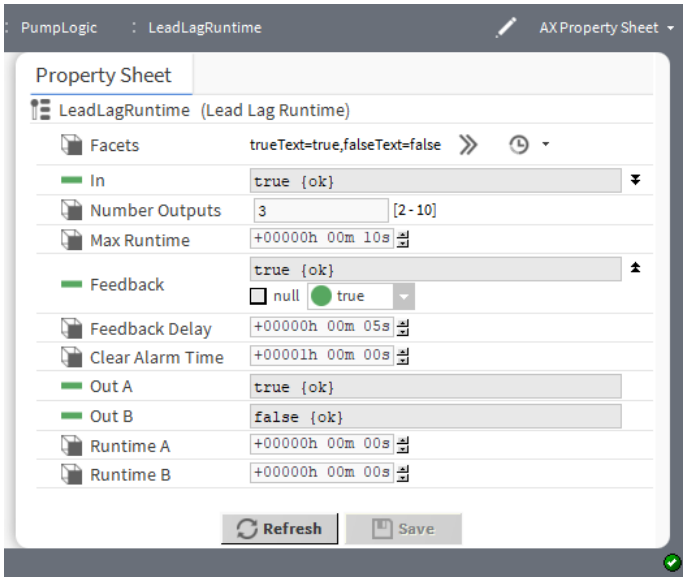
Property	Value	Description
	`false` (default) and null definition	`true` indicates that one of the outputs is active based on the cycle count of each controlled device. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Number Outputs	number from 1–10 (defaults to 2)	Specifies the number of devices (outputs) to control.
Max Runtime	hours, minutes, seconds (defaults to six minutes)	Specifies the maximum amount of time a given output is active (its **`In`** property is set to `true`) before switching to another output.
Feedback	status Boolean input: `true` or `false` (default) and null definition	Provides positive feedback that a controlled device actually started. If the feedback value does not show `true` within the **`Feedback Delay`** time, the current controlled output generates an alarm, and the component switches to the next controlled output. Setting this value to `true` without linking disables this alarm feature. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Feedback Delay	hours, minutes, seconds (defaults to 5 seconds)	Specifies the delay time used to evaluate the feedback link (if any).
Clear alarm time	hours minutes seconds	It clears alarm time.
Out A-J	read only Status-Boolean outputs: `true` or `false` (default)	Each typically links to a BooleanWritable control point with a DiscreteTotalizerExt. Outputs usually control loads of some type, such as two or more pumps.
Cycle Count A-J	integer inputs	These inputs cycle count feedback for the corresponding **`Out A – J`** properties. They typically link to the **`ChangeOfStateCount`** property of the DiscreteTotalizerExt that is measuring the cycles of the corresponding Out A - J properties.

kitControl-LeadLagRuntime

This component provides lead-lag control of from two to 10 BooleanWritables based upon their accumulated runtimes (elapsed active time). This object balances the active runtime of each of the devices. Only one of the controlled devices is active at a time based on runtime

LeadLagRuntime is available in the **HVAC** folder of the **`kitControl`** palette, along with a similar LeadLagCycles object.

Figure 57 LeadLagRuntime properties



Property	Value	Description
Facets	Config Facets window	Defines text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns **`true`**. `falseText` configures the text to describe the state when the component returns **`false`**.
In	status Boolean input: `true` or `false` (default) and null definition	Configures a control device to be on or off. `true` indicates that one of the outputs is active based on the cycle count of each controlled device. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Number Outputs	number from 1–10 (defaults to 2)	Specifies the number of devices (outputs) to control.
Max Runtime	hours, minutes, seconds (defaults to six minutes)	Specifies the maximum amount of time a given output is active (its **`In`** property is set to `true`) before switching to another output.
Feedback	status Boolean input: `true` or `false` (default) and null definition	Provides positive feedback that a controlled device actually started. If the feedback value does not show `true` within the **`Feedback Delay`** time, the current controlled output generates an alarm, and the component switches to the next controlled output. Setting this value to `true` without linking disables this alarm feature.
Feedback Delay	hours, minutes, seconds (defaults to 5 seconds)	Specifies the delay time used to evaluate the feedback link (if any).

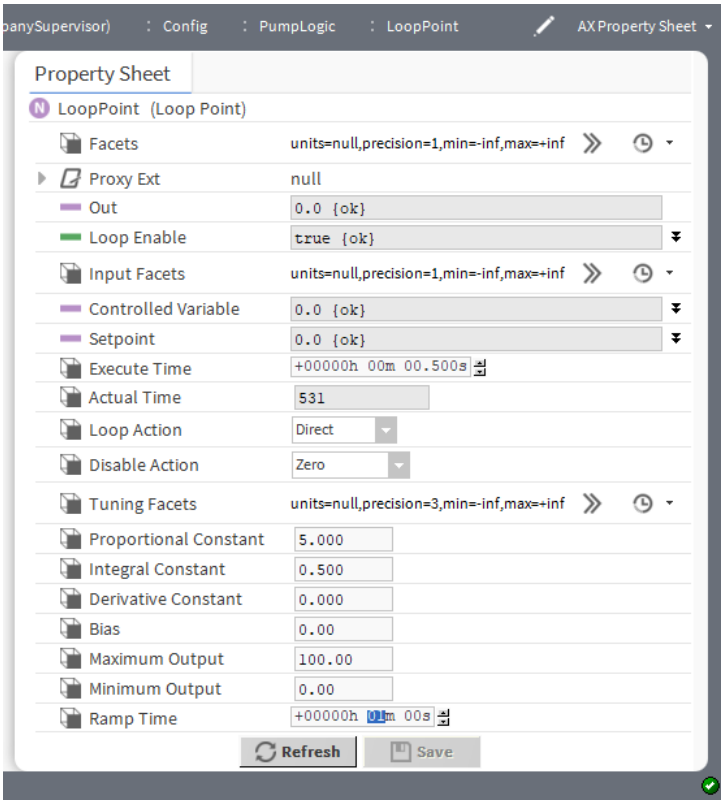
Property	Value	Description
Clear alarm time	hours, minutes, seconds	It clears alarm time.
Out A-J	read-only Status-Boolean outputs: `true` or `false` (default)	Each typically links to a BooleanWritable control point with a DiscreteTotalizerExt. Outputs usually control loads of some type, such as two or more pumps.
Cycle Count A-J	RelTime inputs	These inputs provide runtime feedback for the corresponding **`Out A - J`** properties. They typically link to the **`ElapsedActiveTime`** property of the DiscreteTotalizerExt that is measuring the runtime of the corresponding Out A - J properties.

kitControl-LoopPoint

This component implements a simple PID control loop. Loop objects provide closed-loop PID control (proportional, integral, derivative) at the station level. Independent gain constants configure the loop as P-only, PI, or PID.

LoopPoint is available in the **HVAC** folder of the **`kitControl`** palette.

Figure 58 LoopPoint properties



Property	Value	Description
Facets	Config Facets window	Selects units and configures how the value displays: `unit` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. These facets apply to the **`Out`** value.
Out	read-only number	Displays the output value for the loop.
Loop Enable	`true` (default) or `false` and null definition	Turns the loop on and off. `true` enables the PID loop algorithm to execute at the rate selected by the **`Execute Time`** property. `false` forces the PID loop output to a value that is dependent on the selection in the `Disable Action` property. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Input Facets	Config Facets window	Selects units and configures how the value displays: `unit` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. These facets apply to the input.
Controller Variable	number (defaults to 0.0) and null definition	Defines the input value for the controlled property (for example, space temperature). This input must be valid for this object to function. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Setpoint	number of degrees of temperature and null definition	Defines the input value for the setpoint (for example, the space temperature setpoint). This value must be valid for this object to function. The object does not provide an integral command function for the setpoint value when entered on the property sheet. If a commandable setpoint is required, link from a Numeric-Writable control point to the setpoint slot.
Execute Time	hours minutes seconds (defaults to .5 second)	Controls the execution frequency for the PID algorithm.

Property	Value	Description
Actual Time	read-only integer (defaults to zero (0))	Reports the milliseconds since the previous execution. Changing the **`Execute Time`** changes this time.
Loop Action	drop-down list (defaults to `Direct`)	Determines if the control algorithm is direct or reverse acting. `Direct` increases the loop output as the value of the controlled variable increases to greater than the setpoint value. In a temperature loop, this is considered to be a cooling application. `Reverse` increases the loop output as the value of the controlled variable decreases to less than the setpoint value. In a temperature loop, this is considered to be a heating application.
Disable Action	drop-down list (defaults to `Zero`)	Selects the value for the loop output (**`Out`**) when **`Loop Enable`** is `false`. `Max Value` sets the loop output value to the value configured for **`Max Output`**. `Min Value` sets the loop output value to the value configured for **`Min Output`**. `Hold` maintains the loop output value at the last calculated value. `Zero` sets the loop output value to zero (0.00).
Tuning Facets	Config Facets window	Selects units and configures how the value displays: `unit` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. These facets apply to the tuning properties, such as the constants, **`Bias`**, and output properties.
Proportional Constant	number of seconds to three decimal places	Defines a value used by the loop algorithm to set the overall gain for the loop. A starting point for this value equals: output range/throttling range.
Integral Constant	number to three decimal places	Defines the integral gain in repeats per minute. The loop algorithm uses this value (also called the reset rate) to act on the magnitude of the setpoint error. A typical starting point is 0.5.
Derivative Constant	number to three decimal places	Defines the derivative gain used by the loop algorithm to act on the rate of change of the setpoint error.
Bias	number to two decimal places	Defines an amount of output the algorithm adds to correct any offset error. The algorithm uses this value normally only with proportional control.
Maximum Output	number to two decimal places	Defines the maximum output value that the loop algorithm can produce.

Property	Value	Description
Minimum Output	number to two decimal places	Defines the minimum output value that the loop algorithm can produce.
Ramp Time	hours minutes seconds (defaults to 0)	Defines the minimum time that the output can ramp completely from **`Minimum Output`** to **`Maximum Output`**, effectively establishing a rate-of-change slope. This component enforces this value at station startup or whenever the LoopPoint transitions from disabled to enabled. Once **`Ramp Time`** expires, it no longer affects the output. This property is intended to prevent the loop from opening a valve or other controlled device to its maximum limit ("slamming") during startup. **NOTE:** The default Ramp Time is 0:00:00, or disabled. To constrain the loop output rate of change when the loop starts or is enabled, enter a reasonable **`Ramp Time`** value.

Action

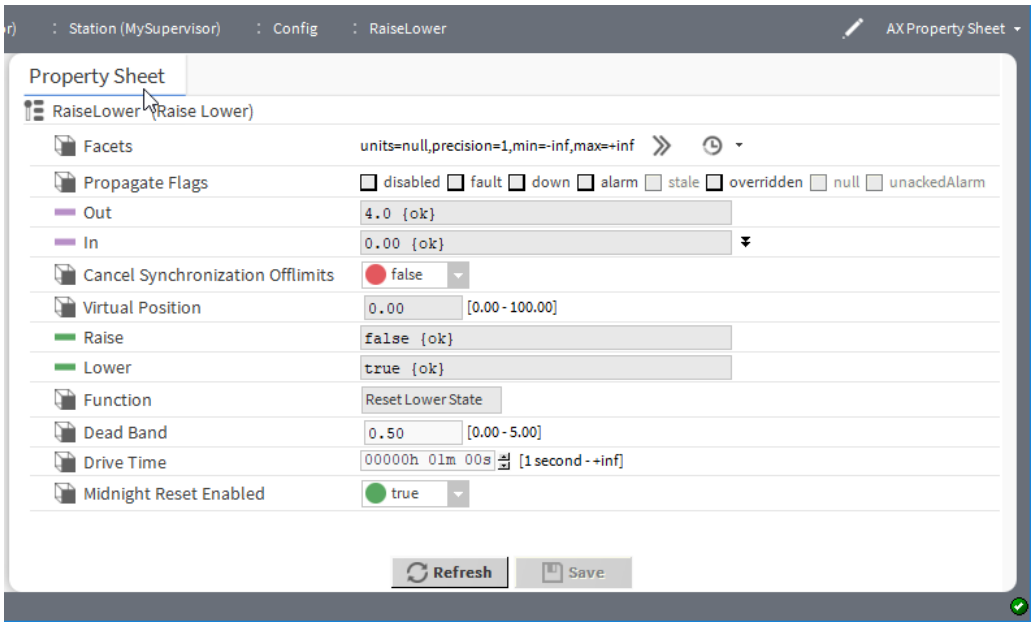
`ResetIntegral` clears the current integral component of the loop's output calculation. If needed, you can link this slot to another object to provide a quick purge of the integral effect. Typically, the latter would provide more of a debug utility, and should not be necessary if the LoopPoint's configuration properties are correctly defined.

kitControl-RaiseLower

This component provides a staged analog output designed to be used with a third party two-relay hardware device. It also provides for the operation of two digital outputs from normal IO hardware such as NDIO as an alternative control method. The actuator should be able to sustain an overdrive at each boundary (for example, a clutch mechanism) as the Raise lower object does not have proportional feedback or limit switch features.

RaiseLower component is available in the **HVAC** folder of the **`kitControl`** palette.

Figure 59 RaiseLower properties



Property	Value	Description
Facets	Config Facets window	Selects units and configures how the value displays: `unit` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This configuration applies to the **`Out`** property value.
Out	read only Status-Numeric number of volts	Displays the analog output value from the object. Valid voltage outputs of 0v, 4v, 7v and 10v depend on the function determined by the object. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
In	status numeric value for percentage from 0 to 100 (defaults to 0.00) and null definition	Defines an input percentage. This slot typically connects to a modulated output of a Control object such as a PID Loop.
Cancel Synchronization Offlimits	`true` or `false` (default)	`true` prevents the doubling of the **`Drive Time`** when moving to the limit positions: 0.0 or 100.0. `false` permits the doubling of **`Drive Time`**.
Virtual Position	status numeric percentage value for percentage from 0 to 100 (defaults to 0.00)	Represents the virtual position of the actuator as calculated by the RaiseLower component.
Raise	status Boolean value: `true` or `false` (default)	`true` configures the output to raise the position of the actuator. The component maintains the raise output for a period of time determined by the positive differential of **`Virtual Position`** subtracted from the **`In`** value, and is a relative proportion of the full-scale drive time. If Virtual Position is 100 percent (fully raised), the component compensates for real time drift by asserting twice the drive time. This synchronizes the physical actuator with the calculated virtual position. `false` leaves the position of the actuator unchanged.
Lower	status Boolean value: `true` or `false` (default)	`true` configures the output to lower the position of the actuator. The component maintains the lower output for a period of time determined by the negative differential of **`Virtual Position`** subtracted from the **`In`** value, and is a relative proportion of the full-scale drive time. If **`Virtual Position`** is 0 percent (fully lowered), the component compensates for real time drift by asserting twice the drive time. This synchronizes the physical actuator with the calculated virtual position. `false` leaves the position of the actuator unchanged.

Property	Value	Description
Function	read-only	Displays operational information corresponding to the current activity of the object. Valid status values include: Off, Lower, Static, Raise.
Dead Band	single digit between 0 and 6 with two decimal places (defaults to 0.50)	Configures an interval where no action occurs and should be set to a value that corresponds to a percentage of full scale drive time. The **In** value must exceed the **Dead Band** value before the **Out** command raises or lowers the actuator.
Drive Time	hours minutes seconds (defaults to one minute)	This should be set to a value that corresponds to the full scale drive time provided by the manufacturer.
Midnight Reset Enabled	`true` (default) or `false`	`true` invokes a synchronization cycle at midnight to compensate for real time drift that may accumulate during normal operation of the actuator. The reset cycle overrides an input signal for a period of twice the full length **Drive Time**. `false` inhibits a reset. This is the desired setting.

kitControl-SequenceBinary

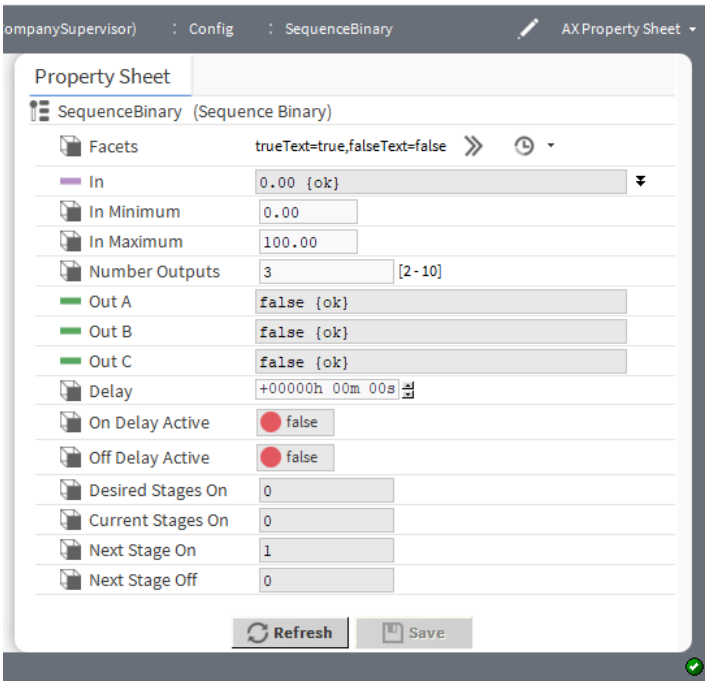
This component provides sequenced weighted staging control of from two to 10 BooleanWritables based upon the status numeric **In** value (0-100). An adjustable delay time is also provided. It can be used to support applications that need to sequence two to 10 loads or stages in a binary sequence. Binary sequencing provides an analog to binary converter function that selects the outputs whose total load rating relates directly to the control need. For each successive output, the output rating is twice the previous output.

SequenceBinary is available in the **HVAC** folder of the **kitControl** palette. A similar object is the SequenceLinear, which uses a rotating method (vs. a weighted method) for sequencing.

For example, this table illustrates how, by controlling three loads, eight unique levels of control can be achieved.

Control Signal (In)%	OutC (4 kw load size)	OutB (2 kw load size)	OutA (1 kw load size)	Stage Hysteresis
100	On	On	On	14.3
87.5	On	On	Off	14.3
71.4	On	Off	On	14.3
57.1	On	Off	Off	14.3
42.9	Off	On	On	14.3
28.6	Off	On	Off	14.3
14.3	Off	Off	On	14.3
0	Off	Off	Off	14.3

Figure 60 SequenceBinary properties



Property	Value	Description
Facets	Config Facets window	This configuration applies to the **Out** property values.
In	number (defaults to 0.00) and null definition	Determines the number of stages that should currently be On. When null is checked, the corresponding value defines the default when a calculation returns null.
In Minimum	single digit to two decimal places (defaults to 0.00)	Defines the **In** value that turns all outputs off.
In Maximum	three-digit number to two decimal places (defaults to 100)	Defines the **In** value that turns all outputs on.
Number Outputs	number between 2 and 10 inclusive (defaults to 3)	Defines the number of outputs or stages.
Out A-J	read-only Status-Boolean values: `true` or `false`	Controls each of from two to 10 loads. The **Number Outputs** property defines how many are available.
Delay	hours minutes seconds (defaults to no delay)	Defines the amount of time that must pass between changes in outputs. The default time is 0 seconds.
On Delay Active	read-only Status-Boolean values: `true` or `false` (default)	Indicates `true` when the on delay timer is active.

Property	Value	Description
Off Delay Active	read-only Status-Boolean values: `true` or `false` (default)	Indicates `true` when the off delay timer is active.
Desired Stages On	read-only number	Indicates the calculated number of stages that should be on based on the **`In`** property.
Current Stages On	read-only number	Indicates the number of stages that are currently on. Normally **`Current Stages On`** and **`Desired Stages On`** are the same. They differ when going through a transition with the delay timer active.
Next Stage On	read-only number	Indicates the next stage to be turned on if needed.
Next Stage Off	read-only number	Indicates the next stage to be turned off if needed.

kitControl-SequenceLinear

This component provides sequenced rotating staging control of from two to 10 BooleanWritables based upon the status numeric **`In`** value (0-100). An adjustable delay time is also provided.

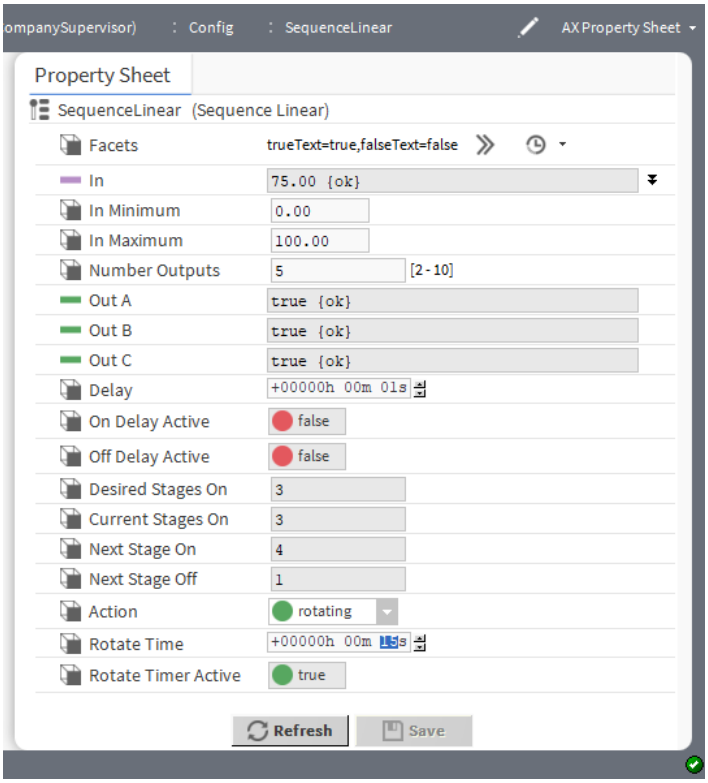
SequenceLinear is available in the **HVAC** folder of the **`kitControl`** palette. A similar object is the SequenceBinary, which uses a weighted method (vs. a rotating method) for sequencing.

This component supports applications that need to sequence two to 10 loads or stages in a linear or rotating sequence. With linear sequencing, the first stage on will be the last stage off. With rotating sequencing, the first stage on will be the first stage off. The **`In`** property, which is a StatusNumeric, controls the number of stages that should be on. The input range is defined by the In Minimum and In Maximum properties.

For example, On and Off setpoints are calculated for each stage by the following formulas (this assumes that five outputs are defined).

	Linear	Rotating
range = InMaximum - InMinimum	100 = 100 - 0	100 = 100 - 0
delta = range / NumberOutputs	20 = 100 / 5	20 = 100 / 5
OnSetpointA = 1 * delta	20	20
OnSetpointB = 2 * delta	40	40
OnSetpointC = 3 * delta	60	60
OnSetpointD = 4 * delta	80	80
OnSetpointE = 5 * delta	100	100
OffSetpointA = 0 * delta, 4 * delta	0	80
OffSetpointB = 1 * delta, 3 * delta	20	60
OffSetpointC = 2 * delta, 2 * delta	40	40
OffSetpointD = 3 * delta, 1 * delta	60	20
OffSetpointE = 4 * delta, 0 * delta	80	0

Figure 61 SequenceBinary properties



Property	Value	Description
Facets	Config Facets window	Defines text to display for each Boolean value. `trueText` configures to describe the state when the component returns**`true`**. `falseText` configures to describe the state when the component returns**`false`**. This configuration applies to the **`Out`** property values.
In	number (defaults to 0.00) and null definition	Determines the number of stages that should currently be On.
In Minimum	single digit to two decimal places (defaults to 0.00)	Defines the **`In`** value that turns all outputs off.
In Maximum	three-digit number to two decimal places (defaults to 100)	Defines the **`In`** value that turns all outputs on.
Number Outputs	number between 2 and 10 inclusive (defaults to 3)	Defines the number of outputs or stages.
Out A-J	read-only Status-Boolean values: `true` or `false`	Controls each of from two to 10 loads. The **`Number Outputs`** property defines how many are available.

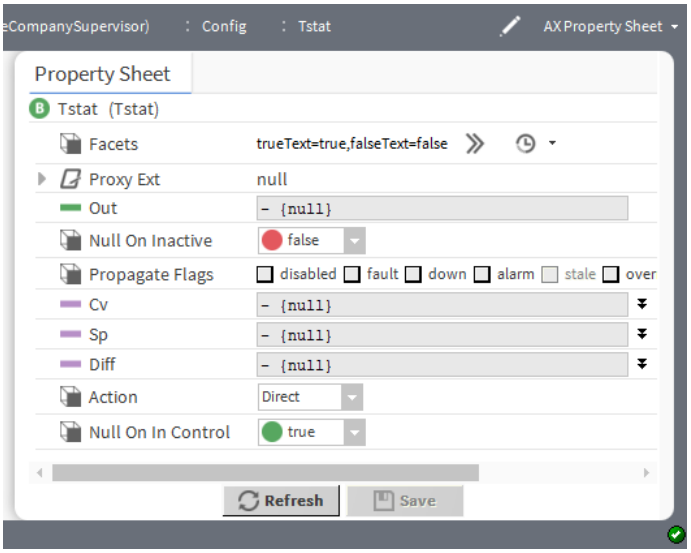
Property	Value	Description
Delay	hours minutes seconds (defaults to no delay)	Defines the amount of time that must pass between changes in outputs. The default time is 0 seconds.
On Delay Active	read-only StatusBoolean values: `true` or `false` (default)	Indicates `true` when the on delay timer is active.
Off Delay Active	read-only StatusBoolean values: `true` or `false` (default)	Indicates `true` when the off delay timer is active.
Desired Stages On	read-only	Indicates the calculated number of stages that should be on based on the **`In`** property.
Current Stages On	read-only	Indicates the number of stages that are currently on. Normally **`Current Stages On`** and **`Desired Stages On`** are the same. They differ when going through a transition with the delay timer active.
Next Stage On	read-only	Indicates the next stage to be turned on if needed.
Next Stage Off	read-only	Indicates the next stage to be turned off if needed.
Action	drop-down list	Selects between Linear and Rotating action. `Linear` configures Out A (Stage 1) to always be the first stage to turn on and the last stage to turn off. `Rotating` configures the first stage to turn on and increment to the next stage each time **`Current Stages On`** goes to 0.
Rotate Time	hours minutes seconds (defaults to one second)	Specifies the amount of time that the outputs remain in a fixed configuration before the component shifts them to the next configuration.
Rotate Timer Active	read-only status Boolean values: `true` or `false` (default)	Indicates that the rotate timer is active.

kitControl-Tstat

This component provides basic thermostatic (On/Off) control with a StatusBoolean Out property and StatusNumeric inputs for controlled variable (Cv), setpoint (Sp), and differential (Diff). An Action property configures the operation directly or in reverse. A **`Null On Inactive`** property is also available. Default action is Direct (cooling).

Tstat is available in the **HVAC** folder of the **`kitControl`** palette.

Figure 62 Tstat properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	This configuration applies to the **`Out`** property value.
Out	read-only Status-Boolean `true` and `false`	Displays the result of the calculation.
Null On Inactive	`true` or `false` (default)	`true` sets the **`Out`** value to null when the component is inactive. `false` retains the last known setting.
Cv	number (defaults to null) and null definition	Defines an input Control Variable (Cv). When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Sp	number (defaults to null) and null definition	Defines the input thermostat Setpoint (Sp). When null is checked, the corresponding value defines the default when a calculation returns null.
Dif	number (defaults to null) and null definition	Defines an input differential (Dif). When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Action	drop-down list	`Direct` performs the calculation directly on the input variables. `Reverse` reverses the calculation.
Null On In Control	`true` (default) or `false`	`true` sets an **`In`** value to null. `false` retains the last known value.

Chapter 8 Latch components

Topics covered in this chapter

♦ kitControl-BooleanLatch
♦ kitControl-EnumLatch
♦ kitControl-NumericLatch
♦ kitControl-StringLatch

Components include services, folders and other model building blocks associated with a module. You may drag them to a property or wire sheet from a palette. These components capture an input value by using either the component's **Clock** property or by using the component's `Latch` action. In either case, latching means setting the value of the latch component's **Out** property to whatever the value of the latch component's **In** property is at the time that the latch occurs. The framework ignores the value of the latch component **In** property unless a latch occurs.

Each latch component type has the same properties and function; they vary only to accommodate different point data types

The descriptions included in the following topics appear as headings in documentation. They also appear as context-sensitive help topics when accessed by:

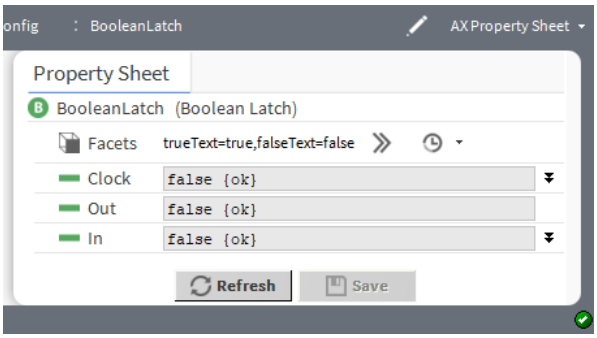
- Right-clicking on the object and selecting **Views→Guide Help**
- Clicking **Help→Guide On Target**

kitControl-BooleanLatch

This component provides a latch for a status Boolean input. It has the same properties and actions as all the latch components.

BooleanLatch is in the **Latches** folder of the **kitControl** palette.

Figure 63 BooleanLatch properties



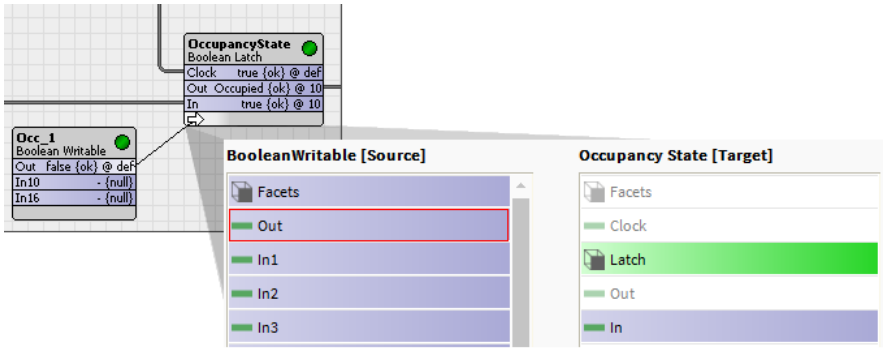
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`. This configuration applies to the **`Out`** property value.
Clock	`true` or `false` (default) and null definition	Triggers the component to capture and send the **`In`** value to the **`Out`** value at the moment this property changes from `false` to `true`. Nothing happens when it changes from `true` to `false`. This is called "latching the input property to the output property on the rising edge." When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.
Out	read-only Status-Boolean `true` or `false` (default)	Contains the value that is captured from the **`In`** property at latch time. You link to this property to display the value on a graphic or to process the value with another component.
In	`true` or `false` (default) and null definition	Contains the value to pass to **`Out`**. You link to this property from a data source, such as from a control point or a Schedule output. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **`In`** value.

Latch action

Latching triggers the component to capture the **`In`** value and send it to the **`Out`** value. In addition to the Clock property's role in triggering a latch, you invoke this action:

- Manually, by right-clicking the latch component and clicking **Actions→Latch**.
- Automatically, by linking a Boolean value to the latch action slot on a Latch component. Any change of the Boolean status value invokes a latch action (`false` to `true` or `true` to `false`). This is in contrast to using the **`Clock`** property, which causes the component to latch only when its status changes from `false` to `true`.

Figure 64 Latch invoked by linking to the latch action



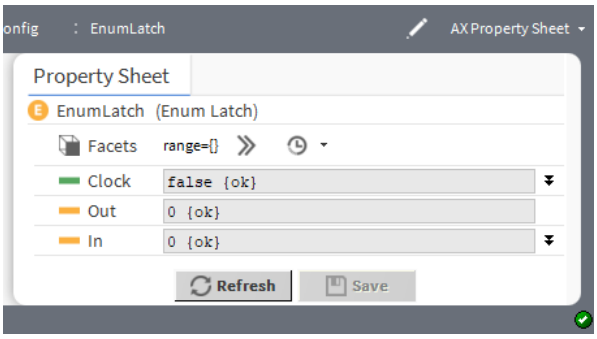
The Latch action captures the input value any time that the Latch command is invoked.

kitControl-EnumLatch

This component provides a latch for a status Enum input. It has the same properties and actions as all the latch components.

kitControl-EnumLatch is available in the **Latches** folder of the **kitControl** palette.

Figure 65 EnumLatch properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines a range of enumerated states (enum values).This configuration applies to the **Out** property value.
Clock	`true` and `false` (default) and null definition	Triggers the component to capture and send the **In** value to the **Out** value at the moment this property changes from `false` to `true`. Nothing happens when it changes from `true` to `false`. This is called "latching the input property to the output property on the rising edge." When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.
Out	read-only enumerated value (defaults to 0)	Contains the value that is captured from the **In** property at latch time. You link to this property to display the value on a graphic or to process the value with another component.
In	number (defaults to 0) and null definition	Contains the value to pass to **Out**. You link to this property from a data source, such as from a control point or a Schedule output. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.

Latch action

Latching triggers the component to capture the **In** value and send it to the **Out** value. In addition to the Clock property's role in triggering a latch, you invoke this action:

- Manually, by right-clicking the latch component and clicking **Actions→Latch**.
- Automatically, by linking a Boolean value to the latch action slot on a Latch component. Any change of the Boolean status value invokes a latch action (`false` to `true` or `true` to `false`). This is in contrast to using the **Clock** property, which causes the component to latch only when its status changes from `false` to `true`.

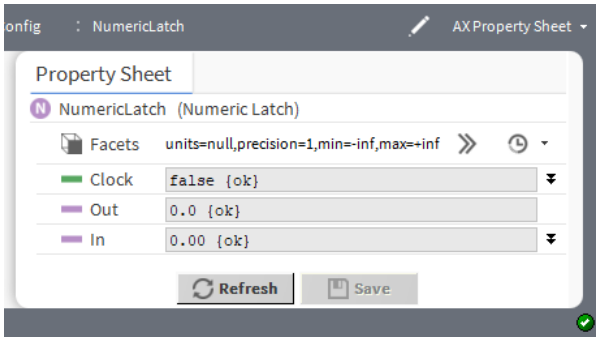
The Latch action captures the input value any time that the Latch command is invoked.

kitControl-NumericLatch

This component provides a latch for a status numeric input.

NumericLatch is available in the **Latches** folder of the **kitControl** palette.

Figure 66 NumericLatch properties



Property	Value	Description
Facets (Numeric)	Config Facets window	Selects units and configures how the value displays: `unit` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. This configuration applies to the **Out** property value.
Clock	`true` and `false` (default) and null definition	Triggers the component to capture and send the **In** value to the **Out** value at the moment this property changes from `false` to `true`. Nothing happens when it changes from `true` to `false`. This is called "latching the input property to the output property on the rising edge." When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.
Out	read-only number (defaults to 0)	Contains the value that is captured from the **In** property at latch time. You link to this property to display the value on a graphic or to process the value with another component.
In	number (defaults to 0) and null definition	Contains the value to pass to **Out**. You link to this property from a data source, such as from a control point or a Schedule output. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.

Latch action

Latching triggers the component to capture the **In** value and send it to the **Out** value. In addition to the Clock property's role in triggering a latch, you invoke this action:

- Manually, by right-clicking the latch component and clicking **Actions→Latch**.
- Automatically, by linking a Boolean value to the latch action slot on a Latch component. Any change of the Boolean status value invokes a latch action (`false` to `true` or `true` to `false`). This is in contrast to

using the **Clock** property, which causes the component to latch only when its status changes from `false` to `true`.

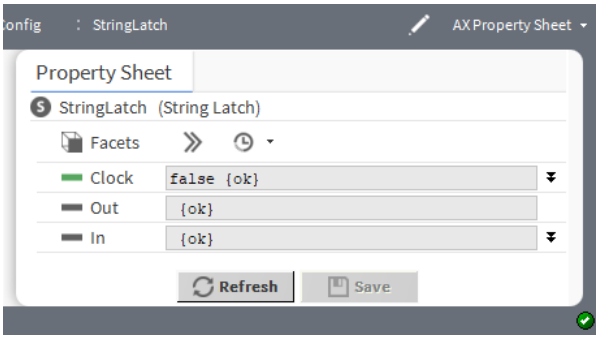
The Latch action captures the input value any time that the Latch command is invoked.

kitControl-StringLatch

This component provides a latch for a StatusString input.

StringLatch is available in the **Latches** folder of the **kitControl** palette.

Figure 67 StringLatch properties



Property	Value	Description
Facets	Config Facets window (defaults to true and false)	Defines the text to display for each Boolean value. `trueText` configures to the text to describe the state when the component returns**true** `falseText` configures to the text to describe the state when the component returns**false** This configuration applies to the **Out** property value.
Clock	`true` and `false` (default) and null definition	Triggers the component to capture and send the **In** value to the **Out** value at the moment this property changes from `false` to `true`. Nothing happens when it changes from `true` to `false`. This is called "latching the input property to the output property on the rising edge." When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.
Out	read-only text (defaults to blank)	Contains the value that is captured from the **In** property at latch time. You link to this property to display the value on a graphic or to process the value with another component.
In	text (defaults to blank) and null definition	Contains the value to pass to **Out**. You link to this property from a data source, such as from a control point or a Schedule output. When null is checked, the value displayed defaults to the incoming value from the device.If you remove the check mark you can configure the **In** value.

Latch action

Latching triggers the component to capture the **In** value and send it to the **Out** value. In addition to the Clock property's role in triggering a latch, you invoke this action:

- Manually, by right-clicking the latch component and clicking **Actions→Latch**.
- Automatically, by linking a Boolean value to the latch action slot on a Latch component. Any change of the Boolean status value invokes a latch action (`false` to `true` or `true` to `false`). This is in contrast to using the **Clock** property, which causes the component to latch only when its status changes from `false` to `true`.

The Latch action captures the input value any time that the Latch command is invoked.

Chapter 9 Logic components

Topics covered in this chapter

♦ kitControl-And
♦ kitControl-Equal
♦ kitControl-GreaterThan
♦ kitControl-GreaterThanEqual
♦ kitControl-LessThan
♦ kitControl-LessThanEqual
♦ kitControl-Not
♦ kitControl-NotEqual
♦ kitControl-Or
♦ kitControl-Xor
♦ kitControl-BqlExprComponent

All 10 of the logic components process input values and provide a StatusBoolean output. Logic object types vary based on input type.

Four logic components receive StatusBoolean inputs:

- And
- Or
- Xor
- Not

Six logic components receive StatusNumeric inputs:

- Equal
- GreaterThan
- GreaterThanEqual
- LessThan
- LessThanEqual
- NotEqual

An ExprLogic component is unlike all others—and technically not a logic component. Instead, it demonstrates how to use an Expr component (BqlExprComponent) with a 4-input logic AND function.

You can add alarm and history extensions to any logic component, in addition to the control extension DiscreteTotalizerExt. If configured with a DiscreteTotalizerExt, you can also add special-purpose alarm extensions ChangeOfStateCountAlarmExt and/or ElapsedActiveTimeAlarmExt.

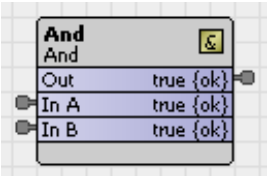
As with math components, you can individually configure logic components to propagate status flags received on linked inputs (by default, status propagation does not occur). For more details, refer to the *Getting Started with Niagara*.

kitControl-And

This component performs a logical AND on all inputs and writes the result to the out property.

kitControl-And is available in the `Logic` folder of the **kitControl** palette.

Figure 68 Logical And with two inputs

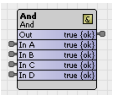


The minimum number of valid inputs for the AND logic object is one. A valid value is one that is “non-null”. In the unique case of only one valid input, the value of the input is passed directly to the output during calculation. The following shows the AND object truth table when using two inputs.

In A	In B	Out
false	false	false
false	true	false
true	false	false
true	true	true

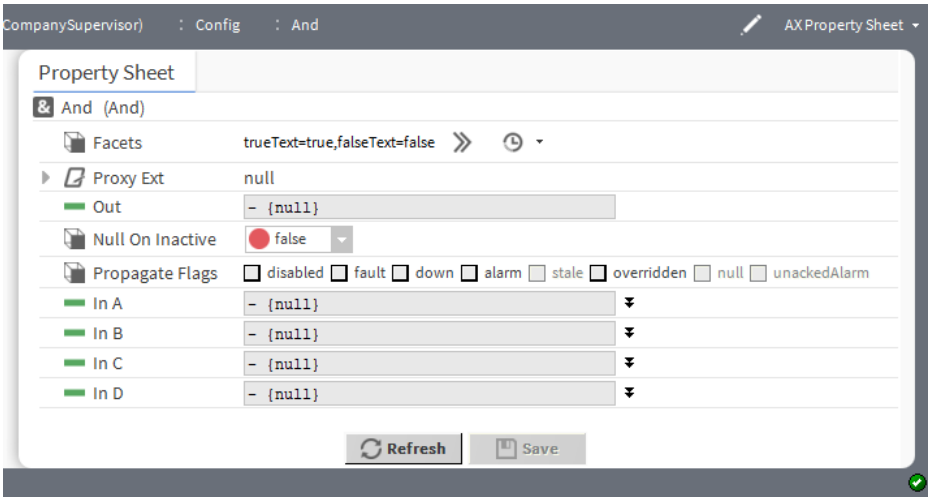
This table shows the object truth using four inputs:

Figure 69 Logical And with four inputs



In A	In B	In C	In D	Out
false	false	false	false	false
false	false	false	true	false
false	false	true	false	false
false	false	true	true	false
false	true	false	false	false
false	true	false	true	false
false	true	true	false	false
false	true	true	true	false
true	false	false	false	false
true	false	false	true	false
true	false	true	false	false
true	false	true	true	false
true	true	false	false	false
true	true	false	true	false
true	true	true	false	false
true	true	true	true	true

Figure 70 And properties



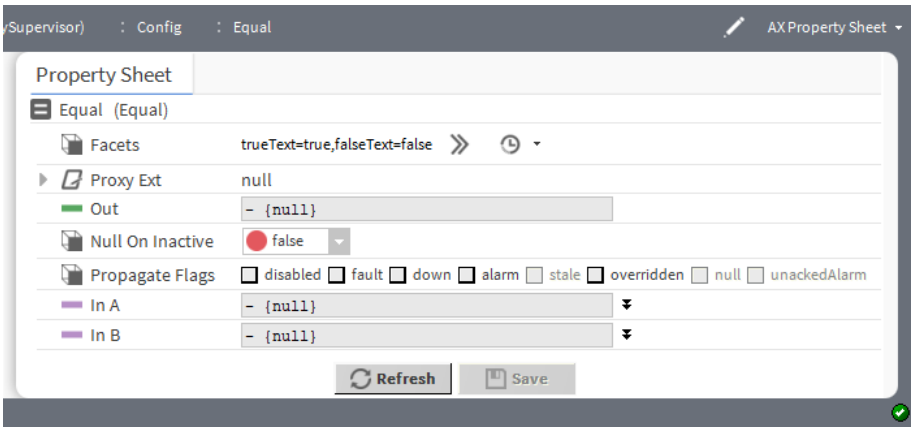
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures to the text to describe the state when the component returns**`true`** `falseText` configures to the text to describe the state when the component returns**`false`**
Out	read-only `true` or `false`	Displays the result of the And operation.
In A through In D	StatusBoolean `true` or `false` and null definition	Provides the input values for the And operation.

kitControl-Equal

This component performs the operation: `A = B`. Numeric.NaN values are never equal.

Equal is available in the **Logic** folder of the **`kitControl`** palette

Figure 71 Equal properties



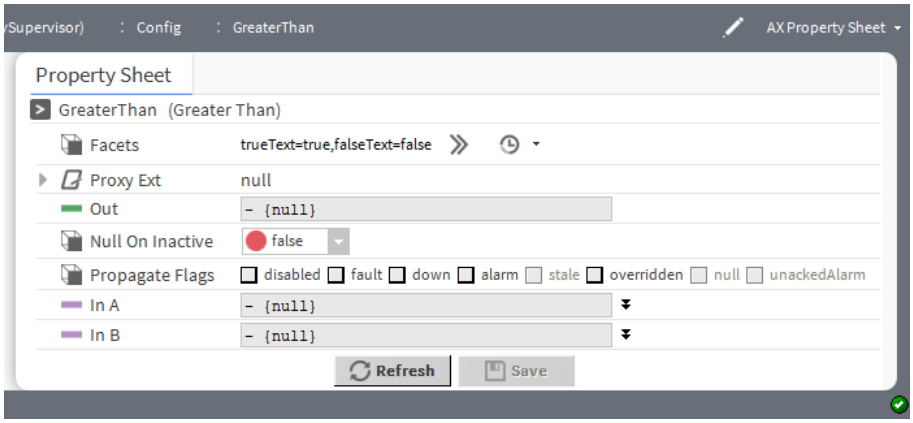
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-GreaterThan

This component performs the operation: `A > B` with a Boolean result.

GreaterThan is available in the **Logic** folder of the **`kitControl`** palette.

Figure 72 GreaterThan properties



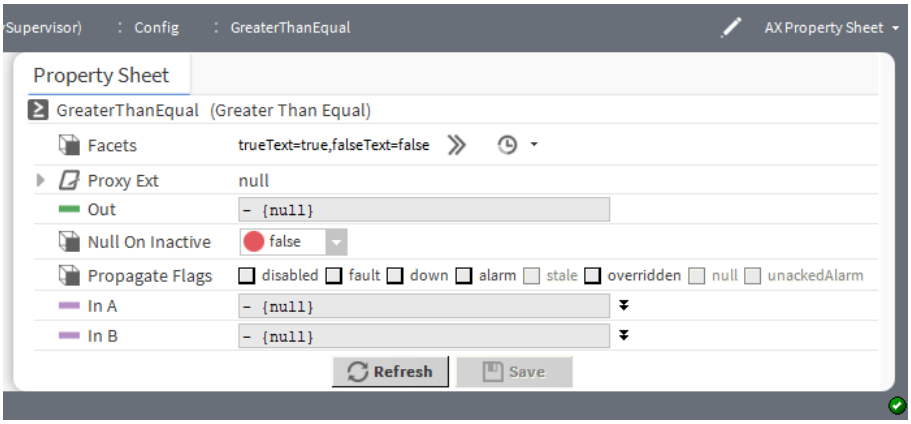
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-GreaterThanEqual

This component performs the operation: `A >= B` with a Boolean result.

GreaterThanEqual is available in the **Logic** folder of the **kitControl** palette

Figure 73 GreaterThanEqual properties



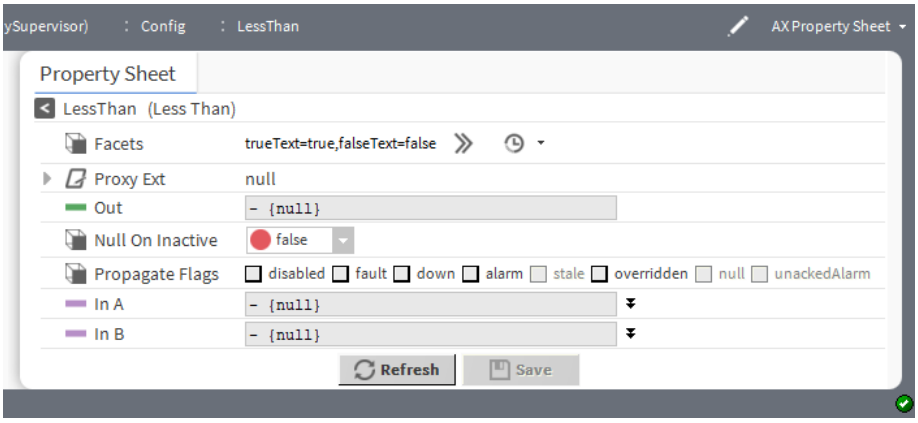
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-LessThan

This component performs the operation: `A < B` with a Boolean result.

LessThan is available in the **Logic** folder of the **kitControl** palette.

Figure 74 LesstThan properties



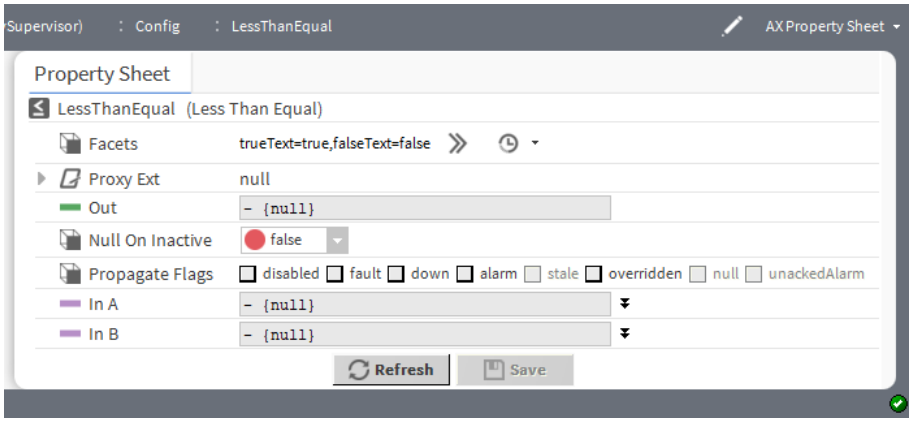
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-LessThanEqual

This component performs the operation: `A <= B` with a boolean result.

LessThanEqual is available in the **Logic** folder of the **`kitControl`** palette.

Figure 75 LessThanEqual properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

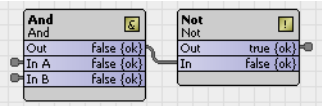
kitControl-Not

This component inverts the Boolean logic value currently at the (single) object input.

Not is available in the **Logic** folder of the **kitControl** palette.

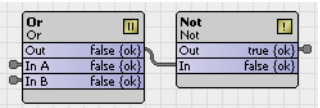
You often link Not objects with other logic objects to make different logic gates. The following tables are examples using Not to create NAND (not and) logic, NOR (not or) logic, and EQUIV (equivalent) gate logic.

Figure 76 NAND logic using And and Not



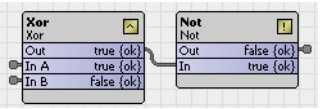
In A	In B	Out
false	false	true
false	true	true
true	false	true
true	true	false

Figure 77 NOR logic using Or and Not



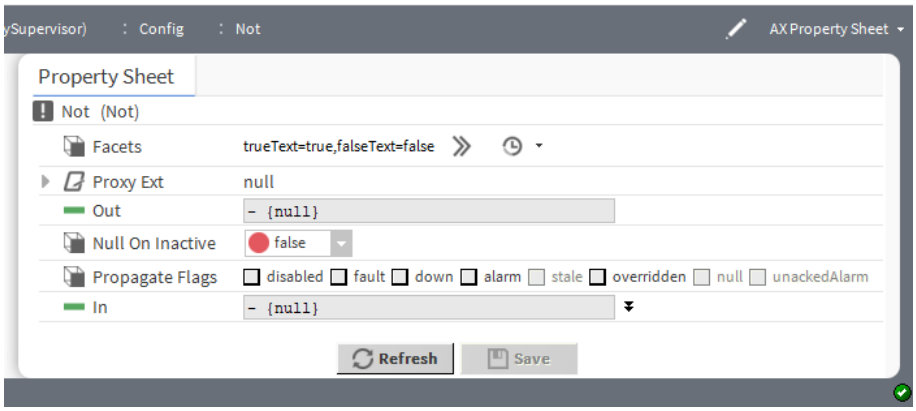
In A	In B	Out
false	false	true
false	true	false
true	false	false
true	true	false

Figure 78 NOR logic using Or and Not



In A	In B	Out
false	false	true
false	true	false
true	false	false
true	true	true

Figure 79 Not properties



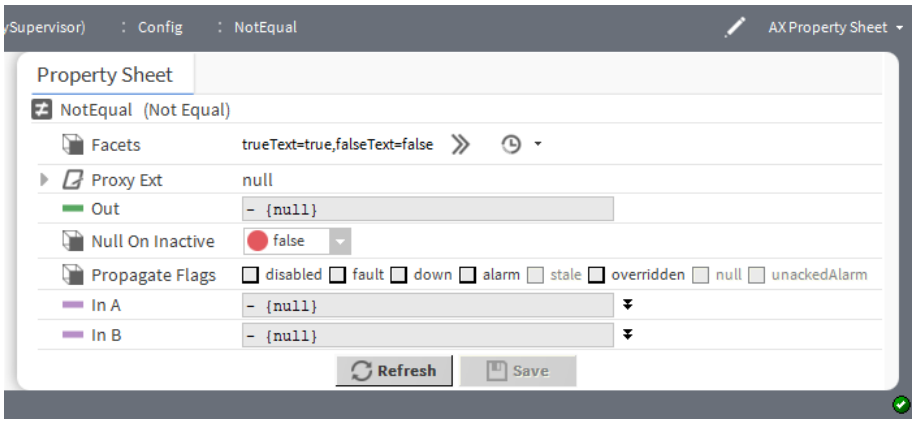
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-NotEqual

This component performs the operation: `A != B` with a Boolean result.

NotEqual is available in the **Logic** folder of the **`kitControl`** palette.

Figure 80 NotEqual properties



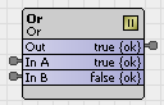
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A and In B	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-Or

This component performs a logical OR on all valid inputs and writes the boolean result to the Out property.

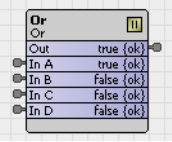
Or is available in the **Logic** folder of the **`kitControl`** palette.

Figure 81 Or object truth table (two inputs)



In A	In B	Out
false	false	false
false	true	true
true	false	true
true	true	true

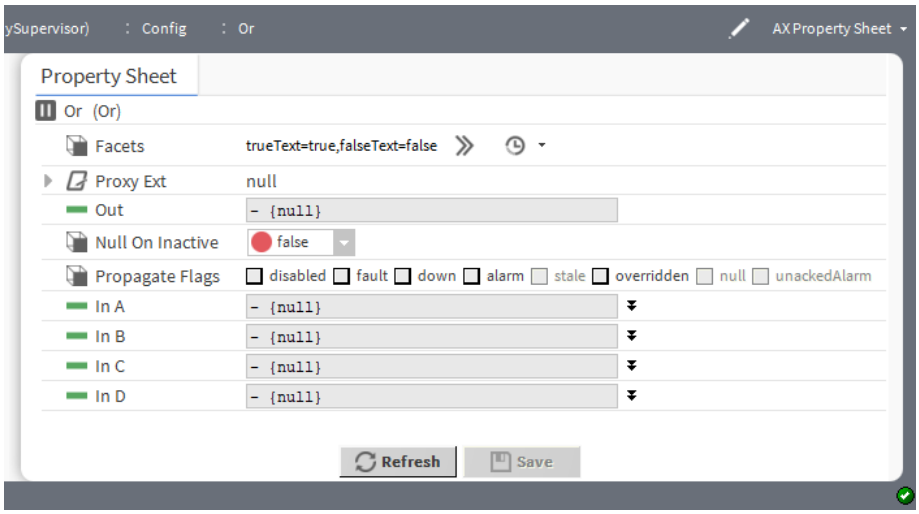
Figure 82 Or object truth table (four inputs)



In A	In B	In C	In D	Out
false	false	false	false	false
false	false	false	true	true
false	false	true	false	true
false	false	true	true	true
false	true	false	false	true
false	true	false	true	true
false	true	true	false	true
false	true	true	true	true
true	false	false	false	true

In A	In B	In C	In D	Out
true	false	false	true	true
true	false	true	false	true
true	false	true	true	true
true	true	false	false	true
true	true	false	true	true
true	true	true	false	true
true	true	true	true	true

Figure 83 Or properties



Property	Value	Description
Facets (Boolean)	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A through In D	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-Xor

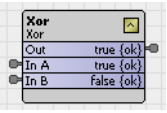
This component performs a logical XOR on all valid inputs and writes the result to the **Out** property.

Xor is available in the **Logic** folder of the **kitControl** palette.

This table shows the Xor object truth table when using two inputs (typical).

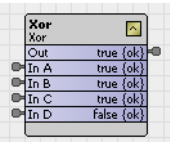
This table shows the Xor object truth table if using all four inputs. EQUIV gate logic is accomplished by linking to a Not object.

Figure 84 Xor object truth table (two inputs)



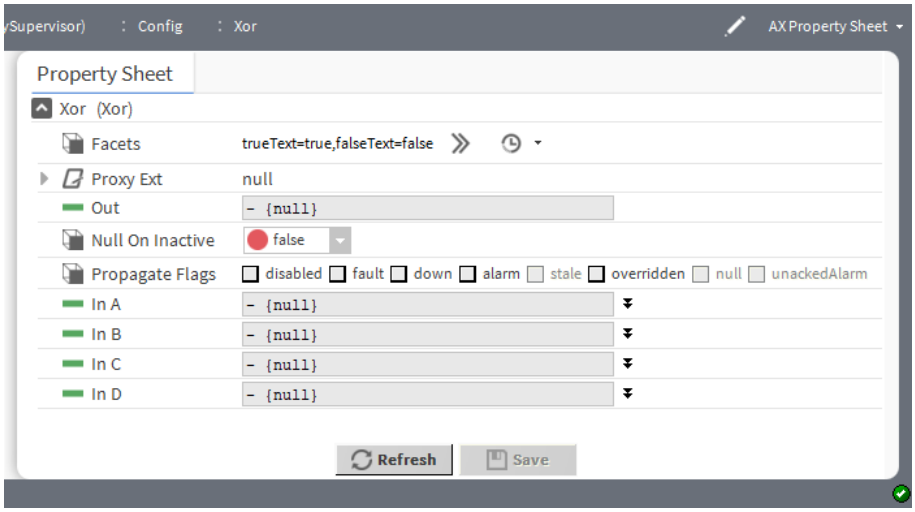
In A	In B	Out
false	false	false
false	true	true
true	false	true
true	true	false

Figure 85 Xor object truth table (four inputs)



In A	In B	In C	In D	Out
false	false	false	false	false
false	false	false	true	true
false	false	true	false	true
false	false	true	true	false
false	true	false	false	true
false	true	false	true	false
false	true	true	false	false
false	true	true	true	true
true	false	false	false	true
true	false	false	true	false
true	false	true	false	false
true	false	true	true	true
true	true	false	false	false
true	true	false	true	true
true	true	true	false	true
true	true	true	true	false

Figure 86 Xor properties



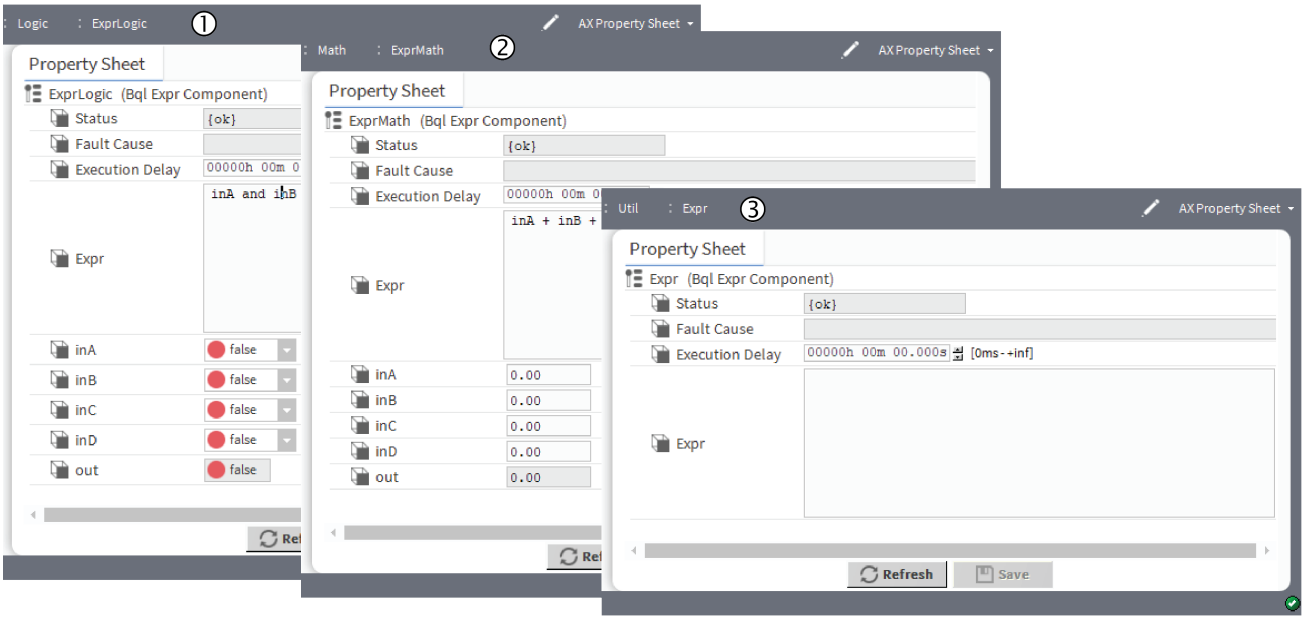
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when component returns `true`. `falseText` configures the text to describe the state when component returns `false`.
Out	read-only `true` or `false`	Displays the result of the comparison.
In A through In D	StatusBoolean `true` or `false` and null definition	Provides the input values for the comparison.

kitControl-BqlExprComponent

This component creates custom math and logic operations based upon manually-added slots and one or more BQL expression statements. Slots can be various baja types, such as primitives Double, Float, Integer, Boolean, or String, or status types: StatusBoolean, StatusNumeric, and so on. Slots are either inputs or one or more outputs. You enter BQL expressions in the component's **`Expr`** (expression) property.

The ExprLogic and ExprMath components are not strictly speaking logic and math components. Instead, they are adaptations of the Expr component (BqlExprComponent), which provide four-input logic and math functions.

Figure 87 The three forms of the BqlExprComponent



- (1) is for logic BQL expressions.
- (2) is for math BQL expressions.
- (3) is for general expressions.

The **Logic** and **Math** folders also contain expression components. ExprLogic provides a four-input logic AND gate. ExprMath provides a four-input math ADD component. The **Util** folder of the **kitControl** palette contains an empty expression component.

For complete information on BQL expressions, refer to Niagara Engineering Notes,*BQL Expression component*.

Property	Value	Description
Status	read only	Indicates the condition of the component at the last check `{ok}` indicates that the component is licensed and polling successfully. `{down}` indicates that the last check was unsucessful, perhaps because of an incorrect property, or possibly loss of network connection. `{disabled}` indicates that the **Enable** property is set to `false`. `false` indicates another problem . Refer to **Fault Cause** for more information.
Fault Cause	read only	Indicates a reason why a system object (network, device, component, extension, etc.) is in fault. This property is empty unless a fault exists.
Execution Delay	hours minutes seconds (defaults to 0)	Configures an amount of time to delay before executing the expression.
Expr	BQL expression	The standard syntax for an expression is as follows: `input operator 'output'`, where:

Property	Value	Description
		input is the name of one or more slots. operator is a word or symbol. 'output' is the slot that contains the result of the expression. (note apostrophes around slot name)
InA, In B, InC and InD	BooleanStatus (`true` or `false` for logic), number (for math)	Defines four input values.
out	read-only Boolean-Status (`true` or `false` for logic), number (for math)	Defines a single output value, which is the result of executing the expression.

The supported operators for the BQL expression are as follows:

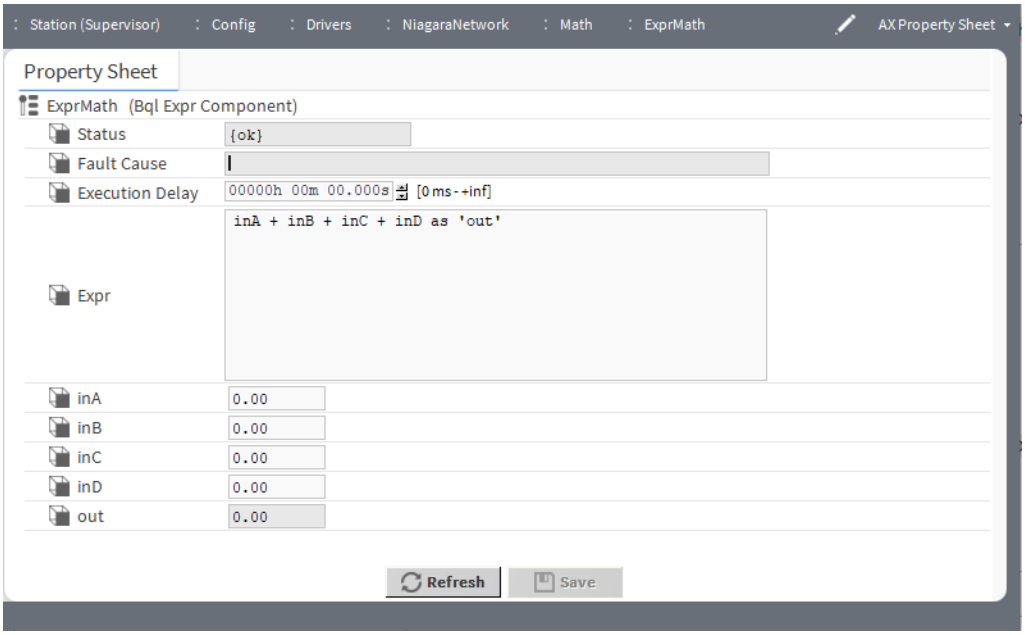
Order of Operation	Operator	Description
1	!, not, -	Logical not, numeric negation
2	* , /	multiplication, division
3	+, -	addition, subtraction
4	=, !=, >, >=, < <=, like	Comparison
5	and, or	Logical operators
6	as	result operator

Operators are processed by their precedence, that is "order of operation", from first (1) to last (6). Parentheses is used to override the normal precedence.

Comma is used as a delimiter to create **`Expr`** with multiple expression(output slots).

For long statements in a **`Expr`** use CR/LF to wrap the text.

BQL Like Query



Following are some examples for Like expression:

- Multiply four Double properties
  inA * inB * inC * inD as ‘out’
- Multiply two BStatusNumeric properties to a Double output property
  inA.value * inB.value as ‘out’
- Negate a Double input property (if inA is 5, out becomes -5)
  -inA as ‘out’

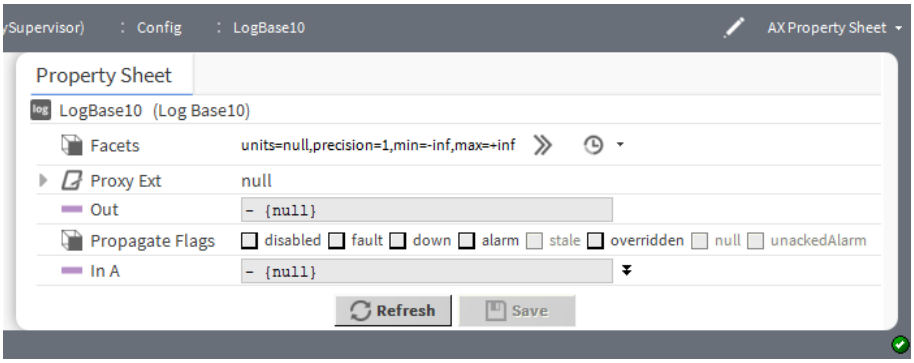
Chapter 10 Math components

Topics covered in this chapter

- kitControl-AbsValue
- kitControl-Add
- kitControl-ArcCosine
- kitControl-ArcSine
- kitControl-ArcTangent
- kitControl-Average
- kitControl-Cosine
- kitControl-Divide
- kitControl-Exponential
- kitControl-Factorial
- kitControl-LogBase10
- kitControl-LogicNatural
- kitControl-Maximum
- kitControl-Minimum
- kitControl-Modulus
- kitControl-Multiply
- kitControl-Negative
- kitControl-Power
- kitControl-Reset
- kitControl-Sine
- kitControl-SquareRoot
- kitControl-Subtract
- kitControl-Tangent
- kitControl-BqlExprComponent

Math components process one or more StatusNumeric input values and provide a StatusNumeric output value. Each component type provides a specific math function, such as Add, Average, Divide, Minimum, Maximum, Reset, AbsValue, and so on.

Figure 88 LogBase10, an example of a single-input math component

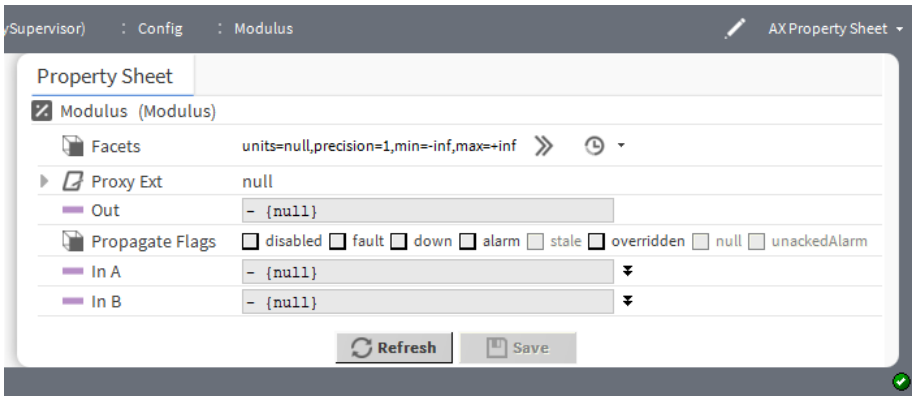


These components perform an operation on a single input:

- AbsValue
- ArcCosing
- ArcSine
- ArcTangent
- Cosine

- Exponential
- Factorial
- LogBase10
- LogNatural
- Negative
- Reset (uses four values of high and low limits, both input and output, also linkable as inputs)
- Sine
- SquareRoot
- Tangent

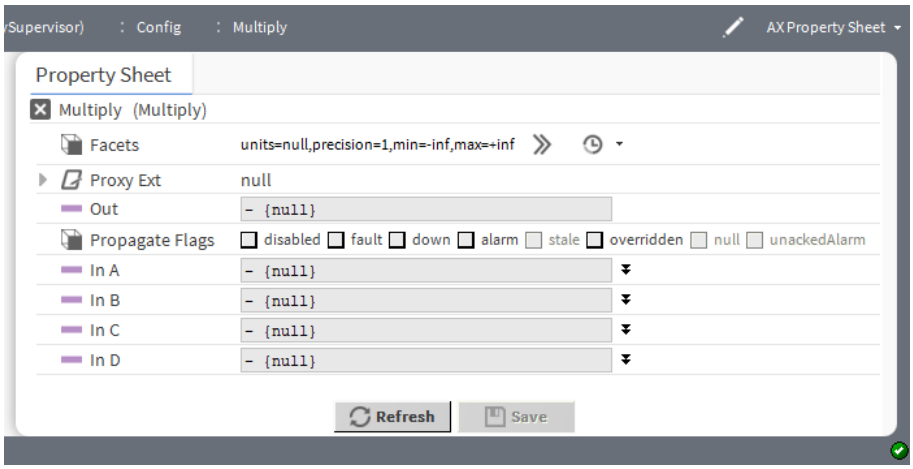
Figure 89 Modulus, an example of a math component with two inputs



These components perform an operation using two inputs:

- Divide
- Modulus
- Power
- Subtract

Figure 90 Modulus, an example of a math component with two inputs



Math object types vary by number of inputs used, in addition to the math operation. These components perform an operation using one to four inputs:

- Add
- Average
- Maximum
- Minimum
- Multiply

If needed, you can add alarm and history extensions to any math component, in addition to the control extension NumericTotalizerExt.

As with logic components, you can individually configure math components to propagate status flags received on linked inputs (by default, status propagation does not occur). For more details refer to the Getting Started Guide.

The properties table summarizes the properties for all math components.

Property	Value	Description
Facets	Config Facets window (defaults to null for units, one decimal place for precision, negative infinity for minimum, and positive infinity for maximum)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`In`** value for this component.
Out	read-only number (defaults to zero (0))	Displays the result of the calculation.
In A In A and In B In A through In D (depends on the operation)	number (defaults to zero (0))	Provides the input values for the calculation.

kitControl-AbsValue

This math component performs the operation: `out = abs(inA)`, in other words, the absolute value of inA.

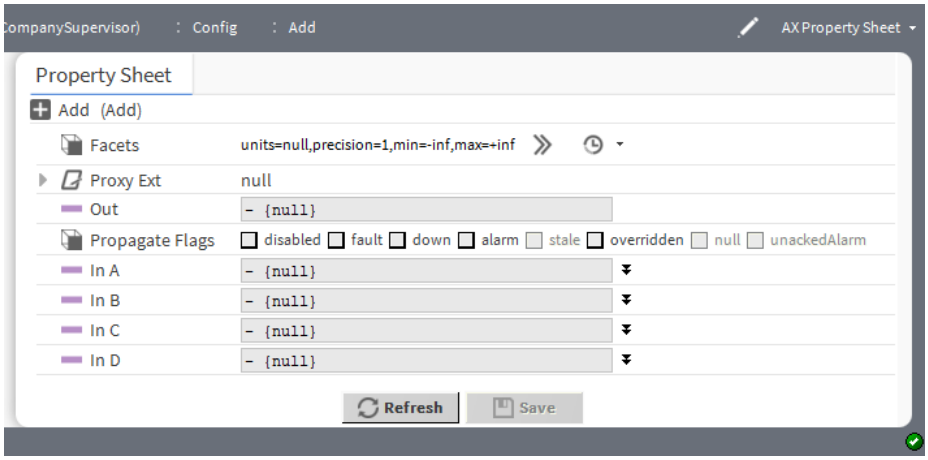
AbsValue is in the **Math** folder of the **`kitControl`** palette. It inputs a value to a widget that is the same as the value output from another widget.

kitControl-Add

This math component performs the operation: `out = (inA + inB + inC + inD)`.

AbsValue is in the **Math** folder of the **`kitControl`** palette. It adds together up to four input values, generating a single output value.

Figure 91 Add properties



Property	Value	Description
Facets (Numeric)	Config Facets window	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places.
Out	read-only number	Displays the sum of the four input values.
In A, In B, In C, In D	numbers (up to four values)	Defines the numbers to add.

kitControl-ArcCosine

This component performs the operation: `out = acos(inA)`.

The ArcCosine is available in the **Math** folder of the **`kitControl`** palette.

kitControl-ArcSine

This component performs the operation: `out = asin(inA)`.

The ArcSine is available in the **Math** folder of the **`kitControl`** palette.

kitControl-ArcTangent

This component performs the operation: `out = atan(inA)`.

The ArcTangent is available in the **Math** folder of the **`kitControl`** palette.

kitControl-Average

This component determines the average value of valid inputs and writes that value to out: `out = (inA + inB + inC + inD) / 4`.

The Average is available in the **Math** folder of the **`kitControl`** palette.

kitControl-Cosine

This component performs the operation: `out = cos(inA)`.

The Cosine is available in the **Math** folder of the **kitControl** palette.

kitControl-Divide

This component performs the operation: `out = (inA / inB)`. If either input is Numeric.NaN, the output will be Numeric.NaN.

The Divide is available in the **Math** folder of the **kitControl** palette.

kitControl-Exponential

This component performs the operation: `out = e ^ inA` (e raised in the inA power).

Exponential is available in the **Math** folder of the **kitControl** palette.

kitControl-Factorial

This component provides a factorial math output, based upon the value present at its statusNumeric input. Only the integer portion of the input value is evaluated—for example, either value of 1.03 or 1.9999 is evaluated as 1.

Factorial is available in the **Math** folder of the **kitControl** palette.

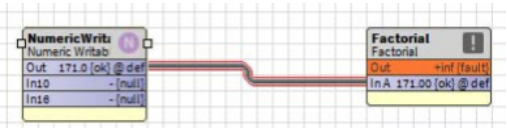
Behavior of Factorial object

This section describes the behavior of the factorial component. If the input value is set to 171 or greater, the output is set to infinite (+inf) with the fault status.

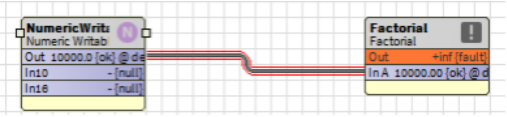
The output displays the message `Input number too large` to the user.

Following are some examples where the inputs are varied:

1. If the **In A** is set to `171`, **Out** becomes `+inf{fault}`.



2. If the **In A** is set to greater than 171, **Out** becomes `+inf{fault}`



.

3. If the **In A** is set to less than 171 (i.e 100), **Out** becomes `1.0 {ok}`.

kitControl-LogBase10

This component performs the operation: `A <= B` with a Boolean result.

LogBase10 is available in the **Logic** folder of the **kitControl** palette.

kitControl-LogicNatural

This component performs the operation: `out = ln(inA) (log base e of inA).`

LogNatural is available in the **Math** folder of the **kitControl** palette.

kitControl-Maximum

This component determines the maximum value of valid inputs and writes that value to the **Out** property. `Out = max (inA, inB, inC, inD).`

Maximum is available in the **Math** folder of the **kitControl** palette

kitControl-Minimum

This component determines the minimum value of valid inputs and writes that value to out. `Out = min (inA, inB, inC, inD).`

Minimum is available in the **Math** folder of the **kitControl** palette.

kitControl-Modulus

This component provides a modulus operation based on values at its two statusNumeric inputs. The output is the remainder of dividing the inA value by the inB value. If the inB value is 0, the output is NaN (not a number). Operation is intended for integer input values, such as from the output of a Counter component.

Modulus is available in the **Math** folder of the **kitControl** palette

kitControl-Multiply

This component performs the calculation: `out = inA * inB * inC * inD`.

Multiply is available in the **Math** folder of the **kitControl** palette.

kitControl-Negative

This component converts any input status numeric to a negative output value.

Negative is available in the **Math** folder of the **kitControl** palette.

kitControl-Power

This component performs the operation: `out = (inA ^ inB)` or a raised to the b power.

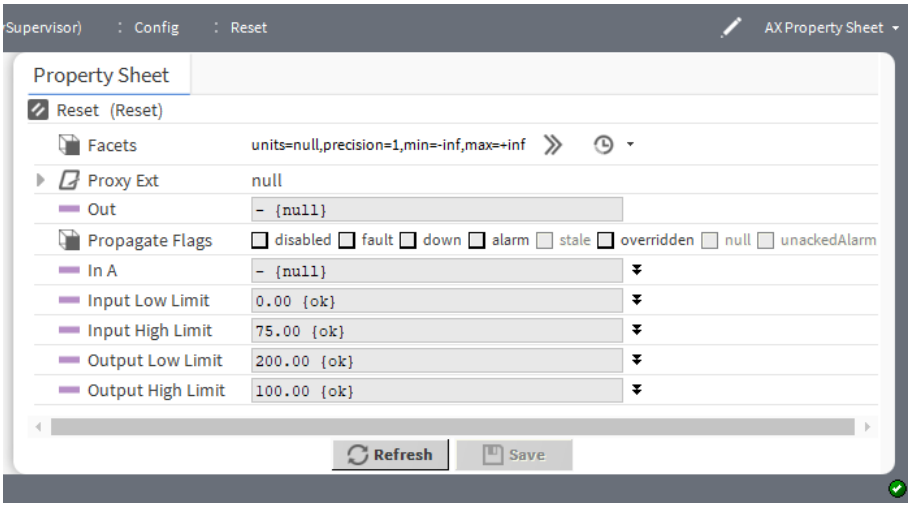
Power is available in the **Math** folder of the **kitControl** palette.

kitControl-Reset

This component performs a linear reset on the **In A** property value.

Reset is available in the **Math** folder of the **kitControl** palette.

Figure 92 Reset properties



Property	Value	Description
Facets	Config Facets window (defaults to null for units, one decimal place for precision, negative infinity for minimum, and positive infinity for maximum)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`In`** value for this component.
Out	read-only number (defaults to zero (0))	Displays the result of the calculation.
In A	degrees of temperature (defaults to null)	Contains one or more input values (degrees of temperature) for the calculation.
Input Low Limit	degrees of temperature (defaults to zero (0))	Sets the lowest limit on outside air temperature. This value must be less than the **`Input High Limit`**.
Input High Limit	degrees of temperature (defaults to zero (0))	Sets the highest limit on outside air temperature. This value must be greater than **`Input Low Limit`**.
Output Low Limit	degrees of temperature (defaults to zero (0))	Sets the lowest limit on water temperature. This value may (or may not) be greater than the **`Output High Limit`**.
Output High Limit	degrees of temperature (defaults to zero (0))	Sets the highest limit on water temperature. This value may (or may not) be greater than the **`Output Low Limit`**.

kitControl-Sine

This component performs the operation: `out = sin(inA)`.

Sine is in the **Math** folder of the **kitControl** palette.

kitControl-SquareRoot

This component performs the operation: `out = sqrt(inA)` (square root of inA).

The SquareRoot is available in the **Math** folder of the **kitControl** palette.

kitControl-Subtract

This component performs the operation: `out = (inA – inB)`. If either input is Numeric.NaN, the output will be Numeric.NaN.

Subtract is available in the **Math** folder of the **kitControl** palette.

kitControl-Tangent

This component performs the operation: `out = tan(inA)`.

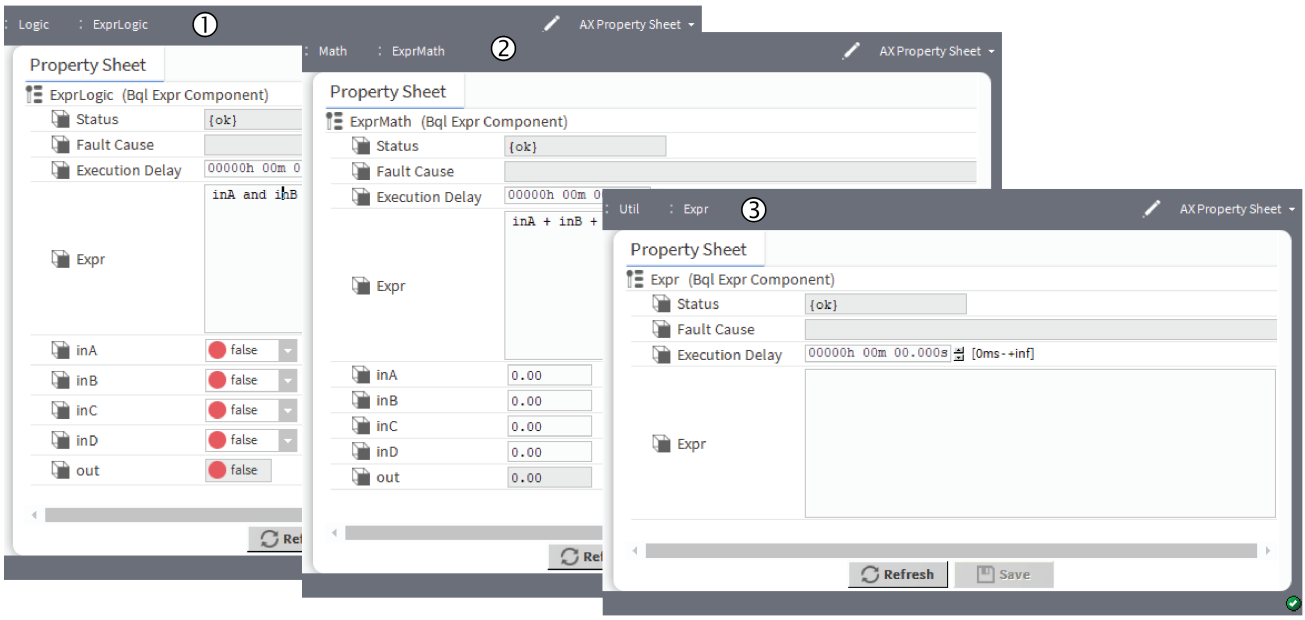
Tangent is available in the **Math** folder of the **kitControl** palette.

kitControl-BqlExprComponent

This component creates custom math and logic operations based upon manually-added slots and one or more BQL expression statements. Slots can be various baja types, such as primitives Double, Float, Integer, Boolean, or String, or status types: StatusBoolean, StatusNumeric, and so on. Slots are either inputs or one or more outputs. You enter BQL expressions in the component's **Expr** (expression) property.

The ExprLogic and ExprMath components are not strictly speaking logic and math components. Instead, they are adaptations of the Expr component (BqlExprComponent), which provide four-input logic and math functions.

Figure 93 The three forms of the BqlExprComponent



- (1) is for logic BQL expressions.
- (2) is for math BQL expressions.

- (3) is for general expressions.

The **Logic** and **Math** folders also contain expression components. ExprLogic provides a four-input logic AND gate. ExprMath provides a four-input math ADD component. The **Util** folder of the **kitControl** palette contains an empty expression component.

For complete information on BQL expressions, refer to Niagara Engineering Notes,*BQL Expression component*.

Property	Value	Description
Status	read only	Indicates the condition of the component at the last check {ok} indicates that the component is licensed and polling successfully. {down} indicates that the last check was unsucessful, perhaps because of an incorrect property, or possibly loss of network connection. {disabled} indicates that the **Enable** property is set to false. false indicates another problem . Refer to **Fault Cause** for more information.
Fault Cause	read only	Indicates a reason why a system object (network, device, component, extension, etc.) is in fault. This property is empty unless a fault exists.
Execution Delay	hours minutes seconds (defaults to 0)	Configures an amount of time to delay before executing the expression.
Expr	BQL expression	The standard syntax for an expression is as follows: input operator 'output', where: input is the name of one or more slots. operator is a word or symbol. 'output' is the slot that contains the result of the expression. (note apostrophes around slot name)
InA, In B, InC and InD	BooleanStatus (true or false for logic), number (for math)	Defines four input values.
out	read-only BooleanStatus (true or false for logic), number (for math)	Defines a single output value, which is the result of executing the expression.

The supported operators for the BQL expression are as follows:

Order of Operation	Operator	Description
1	!, not, -	Logical not, numeric negation
2	* , /	multiplication, division
3	+, -	addition, subtraction
4	=, !=, >, >=, < <=, like	Comparison

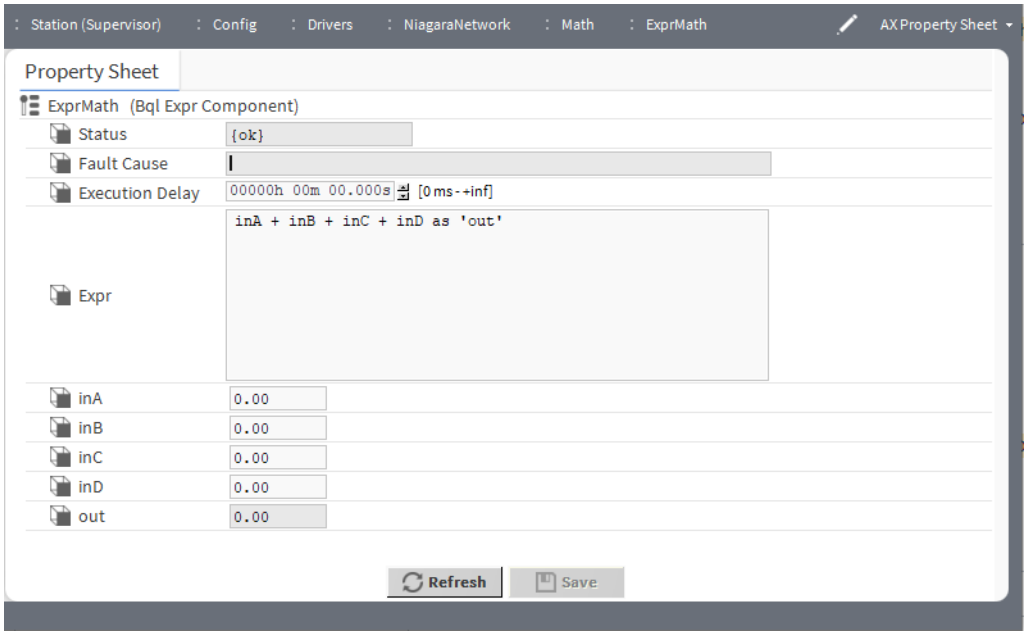
5	and, or	Logical operators
6	as	result operator

Operators are processed by their precedence, that is "order of operation", from first (1) to last (6). Parentheses is used to override the normal precedence.

Comma is used as a delimiter to create **`Expr`** with multiple expression(output slots).

For long statements in a **`Expr`** use CR/LF to wrap the text.

BQL Like Query



Following are some examples for Like expression:

- Multiply four Double properties
  inA * inB * inC * inD as 'out'
- Multiply two BStatusNumeric properties to a Double output property
  inA.value * inB.value as 'out'
- Negate a Double input property (if inA is 5, out becomes -5)
  -inA as 'out'

Chapter 11 Select components

Topics covered in this chapter

♦ kitControl-BooleanSelect
♦ kitControl-EnumSelect
♦ kitControl-NumericSelect
♦ kitControl-StringSelect

Select objects pass one of multiple inputs to the output. You can select from three to 10 inputs for each action.

All select objects require an enumerated input to perform the selection. The four select object types differ only by the type of input data selected and passed to the **Out** property.

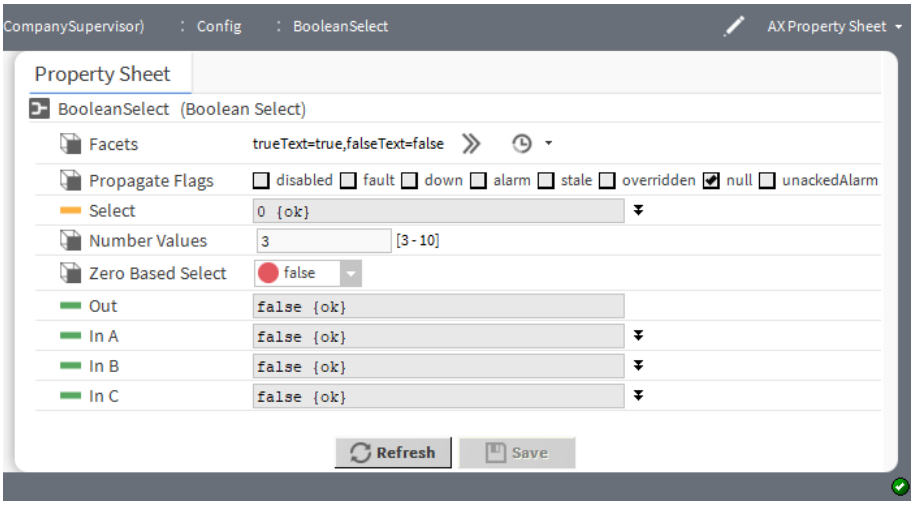
kitControl-BooleanSelect

This component passes one of multiple inputs to a component's **Out** property. The component's **Select** property chooses the input. From three to 10 inputs can be specified.

BooleanSelect is available in the **Select** folder of the **kitControl** palette.

All select objects require an enumerated input to perform the selection—the four select object types differ only by the type of input data selected and passed to the out slot.

Figure 94 BooleanSelect properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`. This configuration affects the **Out** property value.
Select	number (defaults to zero(0))	Selects one of the 10 input values to be transferred to the output.

Property	Value	Description
Out	read-only Status-Boolean `true` or `false` (default)	Provides a Boolean output value for linking into a component. `false` indicates no input values. `true` indicates at least one input value.
In A through In J (10 inputs)	`true` or `false` (default)	Define the selected Boolean input values and null definition.

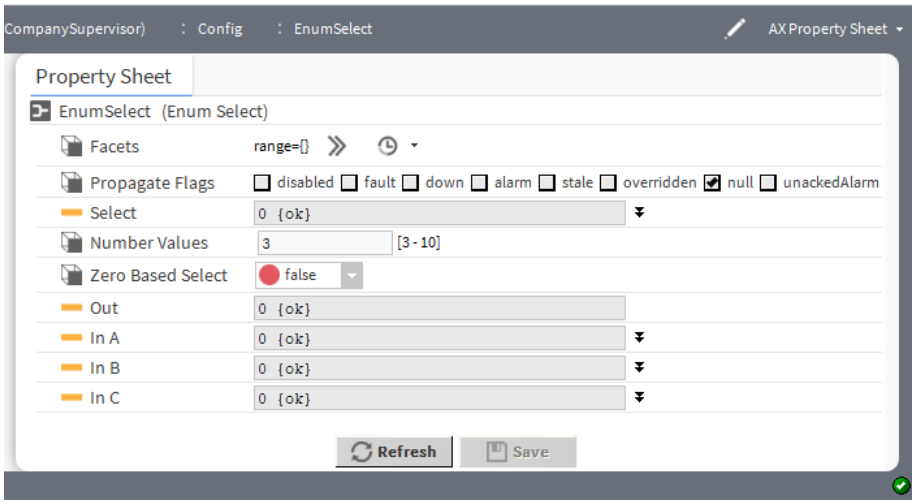
kitControl-EnumSelect

This component passes one of multiple inputs to a component's **`Out`** property. The component's **`Select`** property chooses the input. From three to 10 inputs can be specified.

EnumSelect is available in the **Select** folder of the **`kitControl`** palette.

All select objects require an enumerated input to perform the selection—the four select object types differ only by the type of input data selected and passed to the out slot.

Figure 95 BooleanSelect properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines range of enumerated states (enum values). This configuration affects the **`Out`** property value.
Select	number (defaults to zero (0))	Selects one of the 10 input values to be transferred to the output.
Out	read-only enumerated value (defaults to zero (0))	Provides a numeric output value for linking into a component.
In A through In J (10 inputs)	enumerated value (defaults to zero (0))	Define the selected enum input values and null definition.

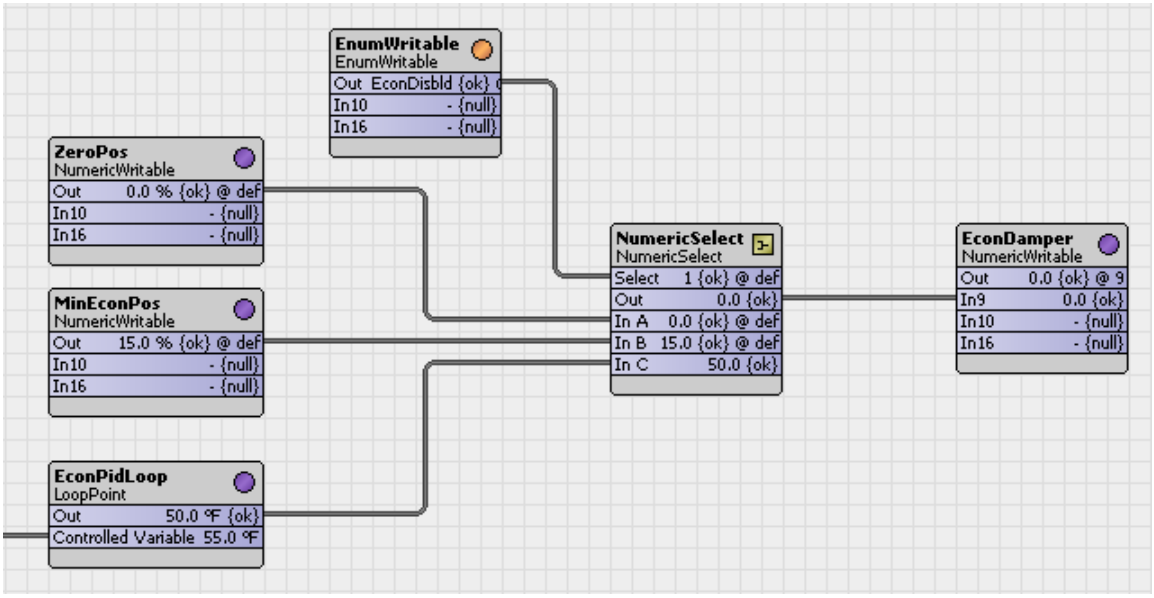
kitControl-NumericSelect

This component is a numeric select. The component's **Select** property chooses the input. From three to 10 inputs can be specified.

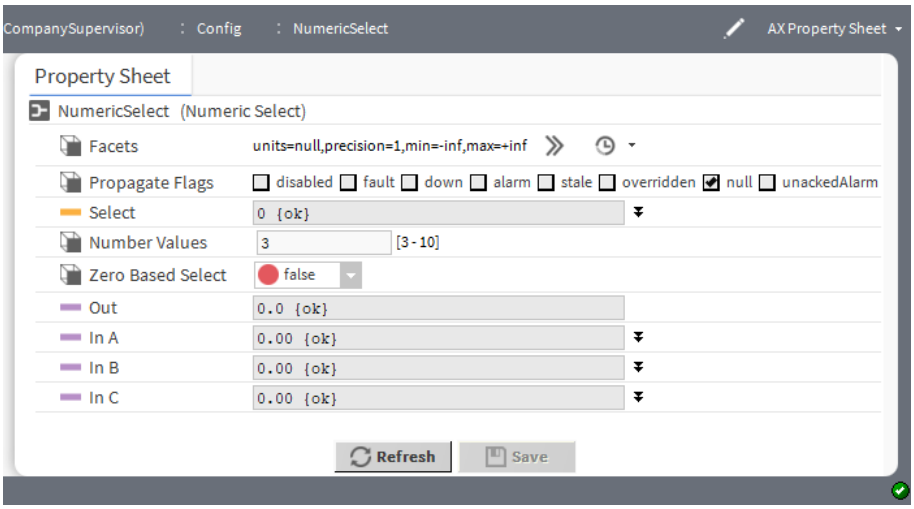
NumericSelect is available in the **Selects** folder of the **kitControl** palette.

All select objects require an enumerated input to perform the selection—the four select object types differ only by the type of input data selected and passed to the out slot.

Figure 96 NumericSelect example application



The wire sheet shows an EnumWritable linked to a NumericSelect's select slot (where enumerated values are 1 = Econ Disabled, 2 = Min Oa Enabled, 3 = Econ Enabled). This sets the output value to one of the input values depending on the select value.

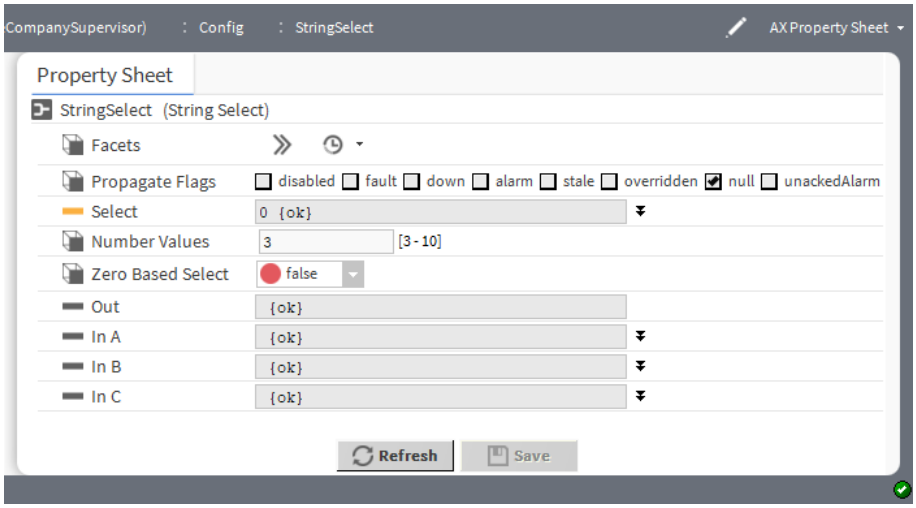


Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum= positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from `acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This configuration affects the **`Out`** property value.
Select	number (defaults to zero(0))	Selects one of the 10 input values to be transferred to the output.
Out	read-only number (defaults to zero (0))	Provides a numeric output value for linking into a component.
In A through In J (10 inputs)	number (defaults to zero (0))	Define the selected numeric input values and null definition.

kitControl-StringSelect

This component is a string select.

StringSelect is available in the **String** folder of the **`kitControl`** palette.



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Associates a key with one or more phrases. This configuration affects the **`Out`** property value.
Select	number (defaults to zero (0))	Selects one of the 10 input values to be transferred to the output.

Property	Value	Description
Out	read-only text (defaults to blank)	Provides a text string output value for linking into a component.
In A through In J (10 inputs)	text (defaults to blank)	Define the selected string input values and null definition.

Chapter 12 String components

Topics covered in this chapter

♦ kitControl-StringConcat
♦ kitControl-StringIndexOf
♦ kitControl-StringLen
♦ kitControl-StringSubstring
♦ kitControl-StringTest
♦ kitControl-StringTrim

These process one or more status string inputs, and produce some form of output.

String object types vary by type of output. Three string components perform a string function with a StatusString output:

- StringConcat
- StringSubstring
- StringTrim

Two string components provide a logic function with a StatusBoolean output:

- StringIndexOf
- StringTest

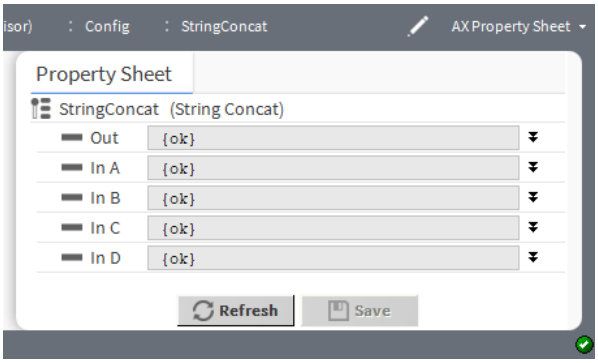
StringLen provides a StatusNumeric output equal to the number of non-null characters in the input string.

kitControl-StringConcat

This component concatenates (joins) up to four StatusString values present on inputs InA, InB, InC, and InD and outputs the concatenated string. String output order is A + B + C + D.

StringConcat is available in the **String** folder of the **kitControl** palette.

Figure 97 StringConcat properties



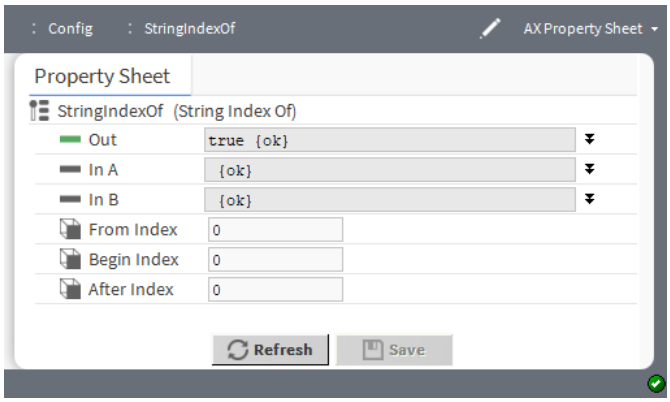
Property	Value	Description
Out	text (defaults to blank) and null definition	Contains the concatenated strings (up to four).
In A through In D	text (defaults to blank) and null definition	Define the strings to concatenate.

kitControl-StringIndexOf

This component compares two StatusString inputs (**In A**, **In B**), looking for an exact match of string **In B** within string **In A**. The object outputs a StatusBoolean result (true, false).

The StringIndexOf is available in the **String** folder of the **kitControl** palette.

Figure 98 StringIndexOf properties



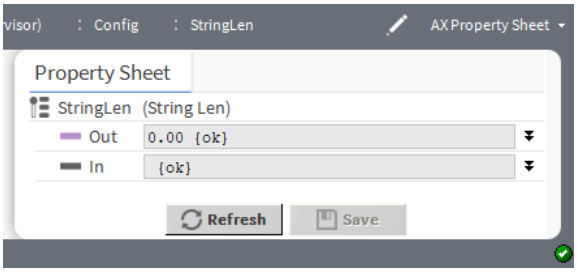
Property	Value	Description
Out	`true` (default) or `false` and null definition	Boolean output indicates the result of the comparison.
In A and In B	text (defaults to blank) and null definition	Define the two strings to compare.
From Index	integer (defaults to 0)	Defines the string from where to start the comparison.
Begin Index	integer (defaults to 0)	Defines which position in the string to start the comparison.
After Index	integer (defaults to 0)	Defines which position in the string to end the comparison.

kitControl-StringLen

This component outputs the total number of non-null characters in the In StatusString value. Out value is StatusNumeric from 0 to *n*.

StringLen is available in the **String** folder of the **kitControl** palette.

Figure 99 StringLen properties



Property	Value	Description
Out	number and null definition	Outputs the length of the string.
In	text (defaults to blank) and null definition	Identifies the string for the length calculation.

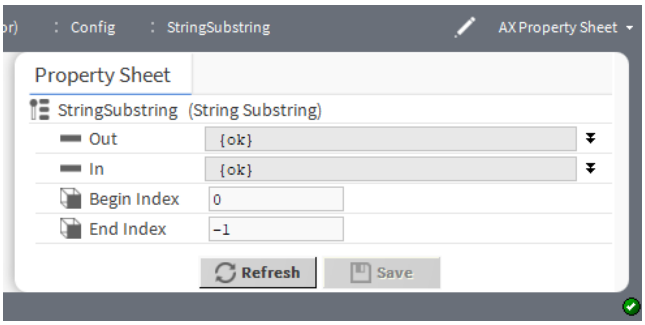
kitControl-StringSubstring

This component outputs a portion of the **In** slot StatusString value, as specified by integer properties **Begin Index** and **End Index**.

StringSubstring is available in the **String** folder of the **kitControl** palette.

By default, **Begin Index**=0 and **End Index**=-1, which means the component passes the entire **In** string.

Figure 100 StringSubstring properties



Property	Value	Description
Out	text (defaults to blank) and null definition	Displays the substring.
In	text (defaults to blank) and null definition	Displays the string from which to extract the substring.
Begin Index	number (defaults to zero (0))	Defines where in the **In** string the substring starts.
End Index	number (defaults to -1)	Defines where in the **In** string the substring ends.

kitControl-StringTest

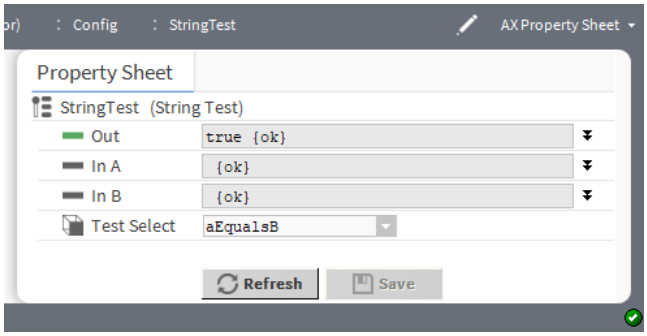
This component compares two StatusString values (**In A** and **In B**) and outputs a StatusBoolean Out (true/false) result.

StringTest is available in the **String** folder of the **kitControl** palette

Test Selection property choices are:

- aEqualsB
- aEqualsBIgnoreCase
- aStartsWithB
- aEndsWithB
- aContainsB

Figure 101 StringTest properties



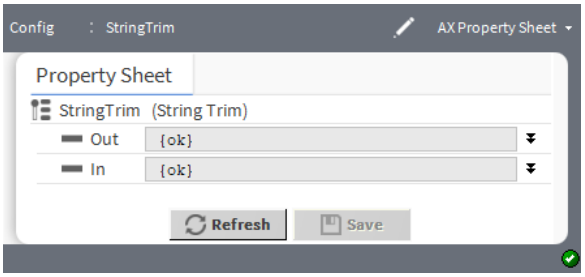
Property	Value	Description
Out	`true` (default) or `false` and null definition	Boolean output indicates the result of the comparison.
In A and In B	text (defaults to blank) and null definition	Define the two strings to compare.
Test Select	drop-down list	Identifies the test to perform on **In A**: `aEqualsB` compares **In A** with **In B** taking case into consideration. `AEqualsBIgnoreCase` compares with **In B** ignoring case. `aStartsWithB` tests to see if **In A** includes **In B** beginning at the first position. `aEndsWithB` tests to see if **In A** includes **In B** beginning at the last position. `aContainsB` tests to see if **In A** includes **In B** anywhere within **In A**.

kitControl-StringTrim

This component trims white space from both ends of the StatusString **In** property value. White space is typically leading or trailing space characters, but may also include control characters.

StringTrim is available in the **String** folder of the **kitControl** palette.

Figure 102 StringTrim properties



Property	Value	Description
Out	text (defaults to blank) and null definition	Outputs the trimmed string.
In	text (defaults to blank) and null definition	Identifies the string from which to strip white space.

Chapter 13 Timer components

Topics covered in this chapter

♦ kitControl-BooleanDelay
♦ kitControl-CurrentTime
♦ kitControl-NumericDelay
♦ kitControl-OneShot
♦ kitControl-TimeDifference

These components include three types of timers, and two components for working with Baja absolute time (AbsTime).

Two of the timers are Boolean-types:

- BooleanDelay
- OneShot

A third dimer is a Numeric-type: NumericDelay.

To work with absolute time, kitControl provides two components:

- CurrentTime
- TimeDifference

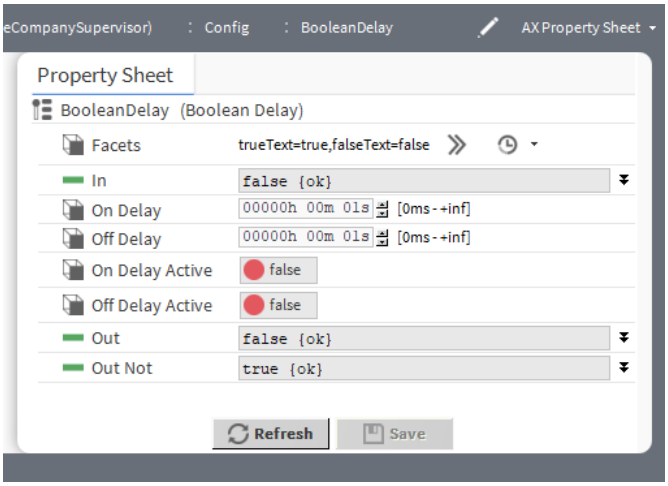
kitControl-BooleanDelay

This component delays the status change of a Boolean status **`Out`** property value by the amount of time configured in the associated **`Delay`** property.

BooleanDelay is available in the **Timer** folder of the **`kitControl`** palette.

Delays relate to on (true) and off (false) statuses and are labeled **`On Delay`** and **`Off Delay`** respectively. The delay applies to any transition (status change from on to off or off to on) at the component's status Boolean input. You configure both delay times in terms of hours, minutes and seconds.

Figure 103 BooleanDelay property sheet



Property	Value	Description
Facets	Config Facets window (defaults to true and false)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`. For example, `trueText` might display `ON` or `Enabled`, and `falseText` `OFF` or `Disabled`.
In	`true` and `false` (default) and null definition	Configures the default input value when no Boolean out value is linked to it. Typically, you link a Boolean Out to this property, in which case, you do not need to manually configure it. You would set this property to `true`, `false` or `null` if no out is linked to it. When this property's status changes, the framework passes this value to the **`Out`** and **`Out Not`** properties after any **`On Delay`** or **`Off Delay`**. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.
On Delay	hours, minutes, seconds (defaults to one second)	Sets the amount of time to wait before sending a `true` value to the **`Out`** property. Time begins to expire the moment the **`In`** property changes from `false` or `null` to `true`.
Off Delay	hours, minutes, seconds (defaults to one second)	Sets the amount of time to wait before sending a false value to the **`Out`** property. Time begins to expire at the moment the **`In`** property changes from `true` to `false` or from `false` to `true` .
On Delay Active	read-only (normally `false`)	Shows if the **`On Delay`** time is actively counting down. This value changes to `true` when the **`In`** property transitions from `false` to `true` and stays unchanged until any **`Off Delay`** time expires. If **`On Delay`** is set to zero (0), this value does not change to `true`.
Off Delay Active	read-only (normally `false`)	Shows if the **`Off Delay`** time is actively counting down. This value changes to `true` when the **`In`** property transitions from `true` to `false` and stays unchanged until any **`Off Delay`** time expires. If **`On Delay`** is set to zero (0), this value does not change to `true`.

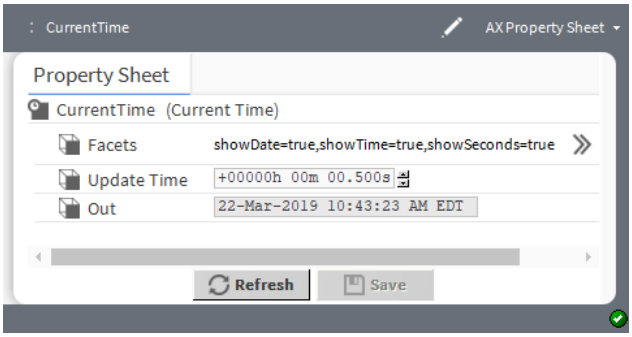
Property	Value	Description
Out	`true` , `false` and null definition	Reflects the **`In`** property value at the end of any **`On Delay`** or **`Off Delay`**. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.
Out Not	`true` , `false` and null definition	Reflects the inverse of the **`In`** property value at the end of any **`On Delay`** or **`Off Delay`**. For example, when the **`In`** value is `true`, the **`Out Not`** value is set to `false` after any delay expires. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**`In`** value.

kitControl-CurrentTime

This component references or displays the current system time formatted in Baja absolute time (AbsTime). You use it directly for graphics display, or with a TimeDifference object for other applications.

CurrentTime is in the **Timer** folder of the **`kitControl`** palette.

Figure 104 CurrentTime properties



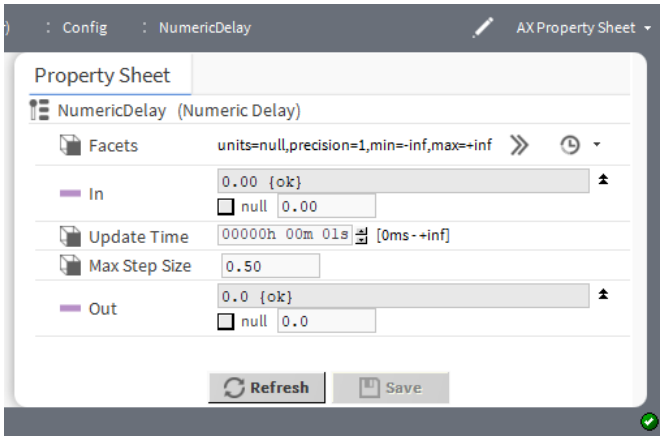
Property	Value	Description
Facets	Config Facets window (defaults to show the date, time and seconds)	Configures how much of the current time to show. For example, some applications may not require time in seconds.
Update Time	hours minutes seconds	Selects the time to output.
Out	read-only current time	Displays the result of the selected time and facets.

kitControl-NumericDelay

This component provides a so called “soft ramp” or “stepped ramp” delay from StatusNumeric **`In`** to **`Out`**. It uses values of **`Max Step Size`** and **`Update Time`** to provide a “stepped” output value. The combination of these two property values determines how quickly and how smoothly the current **`Out`** value changes as it approaches the **`In`** value.

NumericDelay component is located in the **Timer** folder of the **`kitControl`** palette.

Figure 105 NumericDelay properties



Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum= positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** value for this component.
In	number (defaults to 0.00) and null definition	Defines the input value. Typically, you set this value by linking a numeric out value into it. You can manually configure the default state to a numeric value or set it to `null`, so that when no value is linked into this property, the component uses the default value. The component passes this numeric value to the component's **`Out`** property in stages or steps according to the property values in the **`Update Time`** and **`Max Step Size`** properties. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Update Time	hours minutes seconds (defaults to one second)	Determines how often to add the **`Max Step Value`** to the current **`Out`** value. The greater the **`Update Time`** value, the longer it takes for the **`Out`** value to match the **`In`** value. **NOTE:** An Update Time value that is equal to or less than zero (0) disables updating. In this case, the component sets **`In`** , but passes no value to the **`Out`** property.

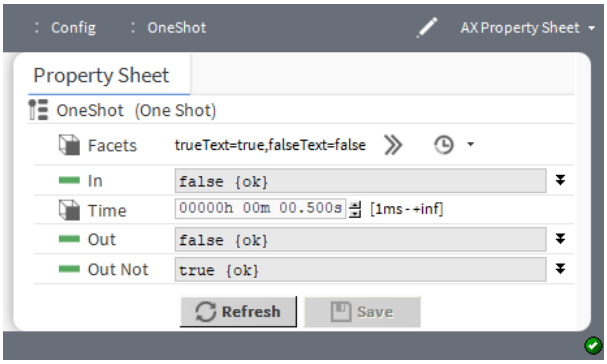
Property	Value	Description
Max Step Size	number to two decimal places (defaults to 0.50)	Limits the amount that may be added with each step that occurs at **Update Time**. If **Update Time** occurs every second, the **Max Step Size** value (or a lesser value) may be added to the current **Out** value every one second until the **Out** value equals the **In** value.
Out	number (defaults to 0.00) and null definition	Displays the current output value as it approaches and equals the **In** property value. The numeric in this property changes at a rate defined by the **Update Time** and **Max Step Size** properties until the value equals the In property value.

kitControl-OneShot

This component provides a single, temporary, status Boolean output for a specified duration (as set in the **Time** property). A OneShot action occurs with a false-to-true value transition at the **In** property, or with an invoked Fire action. When either of these conditions occurs, the component sets the **Out** property value to `true` and the **Out Not** property value to `false` for a time that is equal to the value of the **Time** property. When the time expires, these values revert to the previous default values.

OneShot is available in the **Timer** folder of the **kitControl** palette.

Figure 106 OneShot properties



Property	Value	Description
Facets	Config Facets window	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`. This definition applies to the **Out** value for this component.
In	`true` or `false` (default) and null definition	Defines the input value. You set this property by linking a Boolean **Out** value to it. You can manually configure the default state to a numeric value or set it to `null`, so that when no value is linked into this property, the component uses the default value. The component passes this property value to its **Out** property for the amount of time set in the **Time** property. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

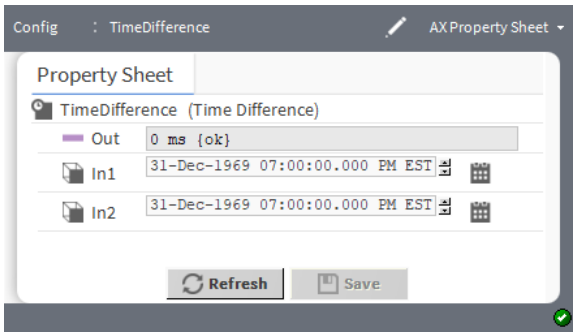
Property	Value	Description
Time	hours minutes seconds (defaults to .5 second)	Determines how long the **Out** and **Out Not** properties hold their one-shot values. For example, a **Time** property value of 2 holds the **Out** property at `true` and the **Out Not** property value at `false` for two seconds.
Out	`true` or `false` (default) and null definition	Displays the current value (display text) that changes with a`false` to `true` transition at the **In** property value or a Fire action. Using the **Facets** property, you configure the **Out** value display text as desired. After a OneShot is triggered and the **Time** value period expires, this value returns to the default (`false`) value. `null` does not change the value with a OneShot Fire action or `false` to `true` transition at the **In** property value. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Out Not	`true` (default) or `false` and null definition	Displays the opposite of the **Out** property. This value changes with a `false` to `true` transition at the **In** property value or a Fire action. After a OneShot is triggered and the **Time** value period expires, this value returns to the default (`true`) value. `null` does not change with a OneShot Fire action or `false` to `true` transition of the **In** property value. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

kitControl-TimeDifference

This component subtracts **In2** from **In1**, and outputs it as a StatusNumeric in milliseconds. You may link a CurrentTime object to one of the inputs for a countdown or count-up type of output.

TimeDifference is in the **Timer** folder of the **kitControl** palette

Figure 107 TimeDifference properties



Property	Value	Description
Out	read-only milliseconds	Displays the numeric value that represents the time difference between **In1** property value and **In2** property value.
In1	date hour:minute: second AM/PM Time zone	Defines a value from which to subtract **In2**.
In2	date hour:minute: second AM/PM Time zone	Defines the value to subtract from **In1**.

Chapter 14 Util components

Topics covered in this chapter

- kitControl-BooleanSwitch
- kitControl-Counter
- kitControl-DigitalInputDemux
- kitControl-EnumSwitch
- kitControl-MinMaxAvg
- kitControl-MultiVibrator
- kitControl-NumericBitAnd
- kitControl-NumericBitOr
- kitControl-NumericBitXor
- kitControl-NumericSwitch
- kitControl-NumericToBitsDemux
- kitControl-Ramp
- kitControl-Random
- kitControl-SineWave
- kitControl-StatusDemux
- kitControl-BqlExprComponent

These components range from a Counter object with Boolean input and a numeric output (counts active transitions) to a Boolean-controlled NumericSwitch that selects one of two numeric values. A StatusDemux provides a method to logic test (true/false) a linked object for the presence of one or more status flags.

Util components also include generator components for simulation/logic testing, such as the SineWave and Ramp (each with a numeric output) and MultiVibrator (Boolean output).

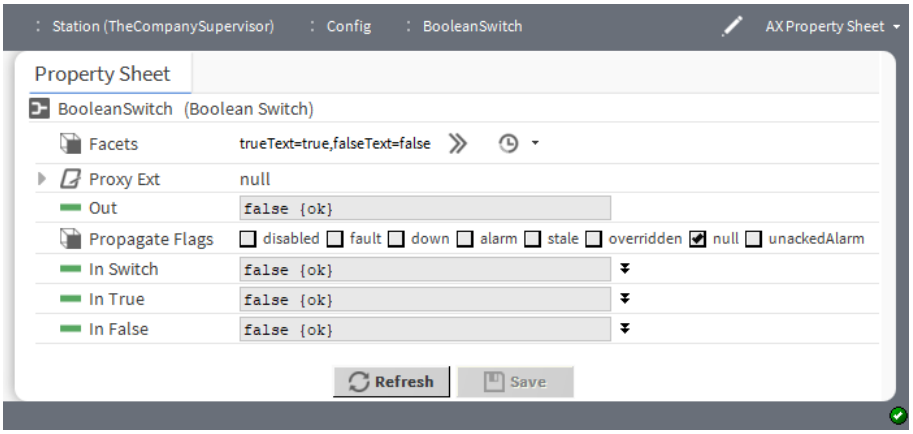
The Expr (BqlExprComponent) component creates custom math and logic functions based upon one or more BQL expression statements.

kitControl-BooleanSwitch

This component selects one of two StatusBoolean inputs based upon the Boolean value at the StatusBoolean input **`In Switch`**.

BooleanSwitch is available in the **Util** folder of the **`kitControl`** palette.

Figure 108 BooleanSwitch properties



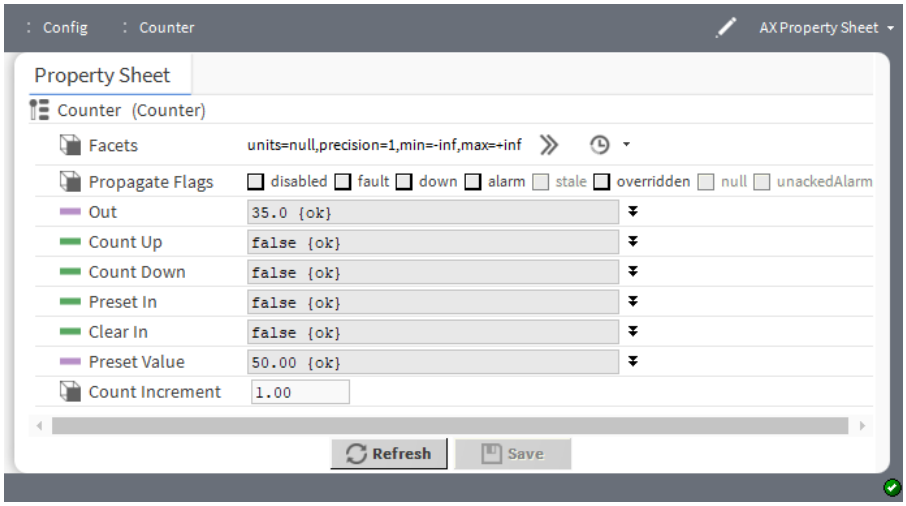
Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`.
Out	read-only Status-Boolean `true` or `false` (default)	Displays the result of the selection.
In Switch	`true` or `false` (default) and null definition	Indicates which input value to select.
In True	`true` or `false` (default) and null definition	Returns `true`.
In False	`true` or `false` (default) and null definition	Returns `false`.

kitControl-Counter

This component counts Boolean inactive to active transitions. It supports counting up, counting down, presetting, and clearing.

The Counter is available in the **Util** folder of the **`kitControl`** palette.

Figure 109 Counter properties



The screen capture shows a counter component property sheet.

Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum to negative infinity, maximum to positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** value for this component.
Out	number (defaults to 0.0) and null definition	Reports the current count. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Count Up	`true` or `false` (default) and null definition	Defines a StatusBoolean input. When this input transitions from inactive to active, the value of the **`Out`** property increments by the **`Count Increment`** value. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Count Down	`true` or `false` (default) and null definition	Defines a StatusBoolean input. When this input transitions from inactive to active, the **`Out`** property decrements by the **`Count Increment`**. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Preset In	`true` or `false` (default) and null definition	Defines a StatusBoolean input. When this input transitions from inactive to active, the **`Out`** property decrements by the **`Count Increment`**. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.
Clear In	`true` or `false` (default) and null definition	Defines a StatusBoolean input. When this input transitions from inactive to active, the framework sets the **`Out`** property to 0.0. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **`In`** value.

Property	Value	Description
Preset Value	number (defaults to 0.00) and null definition	Defines the value of the **Out** property when **Preset In** changes to active, or when a user invokes the Preset action. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.
Count Increment	number (defaults to 1.00) and null definition	Defines how much the **Out** property increases or decreases when a StatusBoolean transitions from inactive to active. When null value is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the **In** value.

Actions

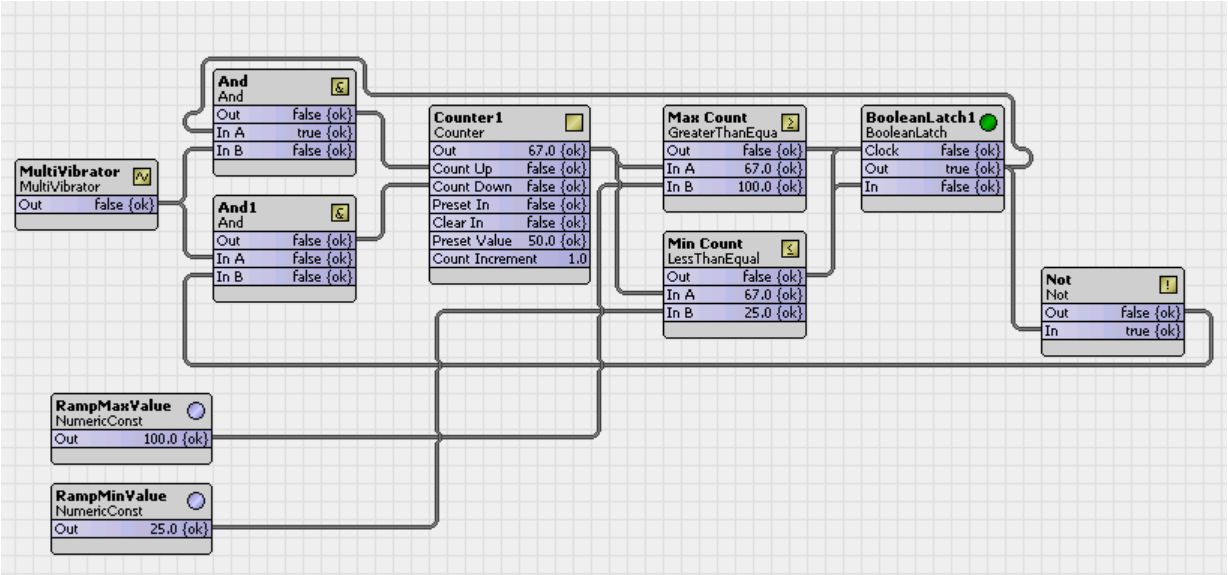
These actions are available when you right-click.

Action	Description
Preset	Sets the **Out** property value to the **Preset Value**.
Clear	Sets the **Out** property value to 0.00.

Examples

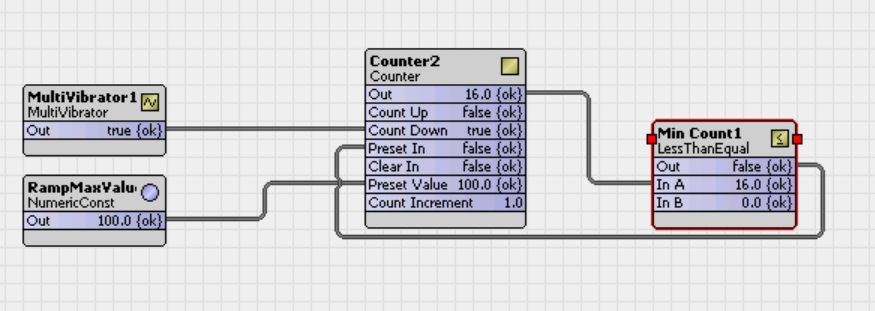
This wire sheet shows an application that ramps between the RampMaxValue and the RampMinValue. The period of the Multi Vibrator object sets how fast the ramp counts. The Clock input of the BooleanLatch object config flags is set to allow fan-in.

Figure 110 Counter component ramp up and down



The next wire sheet shows a count from the RampMaxValue down to zero (0), which is then reset back to the RampMaxValue and repeated.

Figure 111 Counter component ramp down, reset and repeat



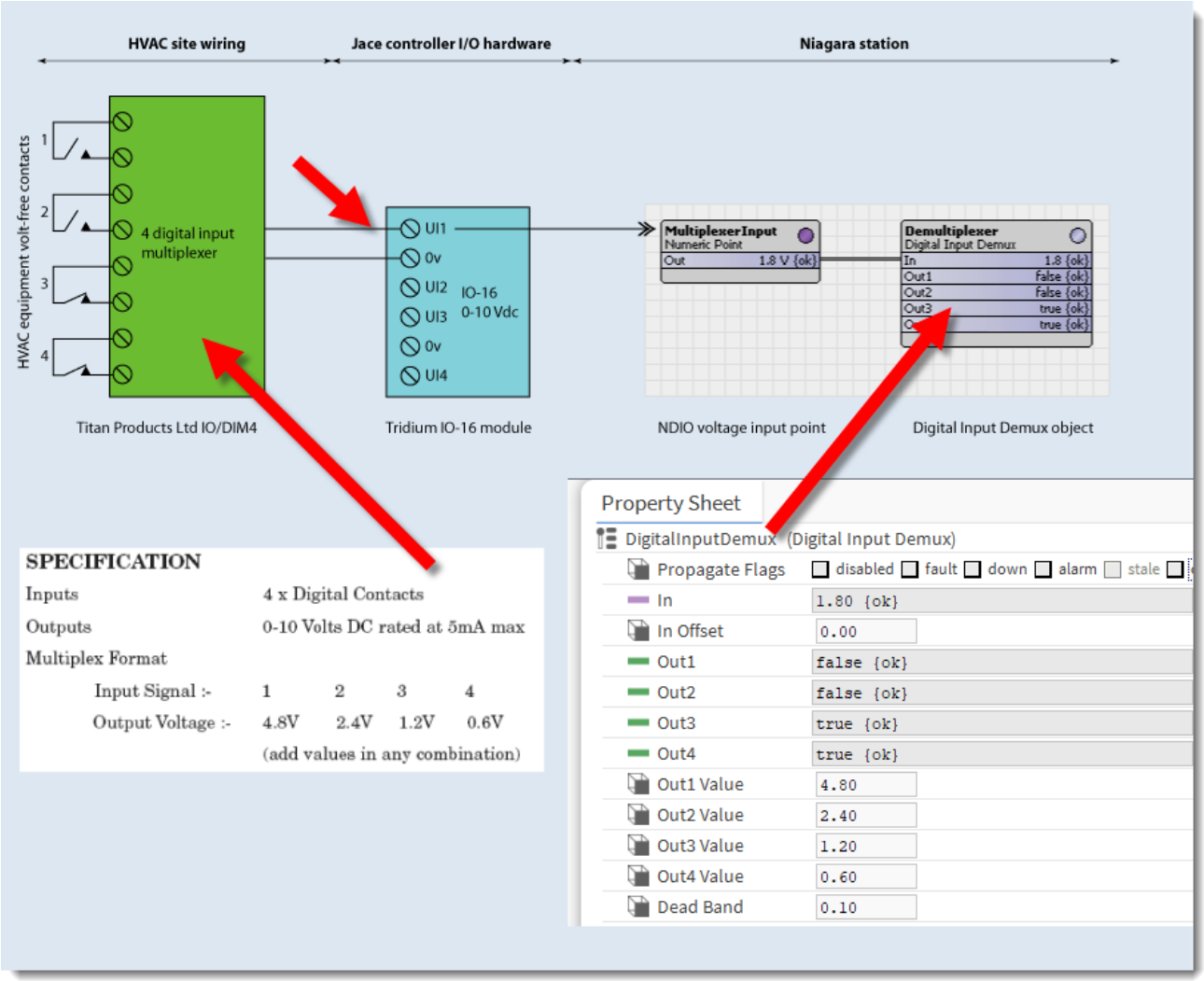
kitControl-DigitalInputDemux

This component provides four StatusBoolean outputs from one StatusNumeric input.

DigitalInputDemux is available in the **Util** folder of the **kitControl** palette.

A typical application for this demultiplexer object is in association with a multiplexer module to expand the IO capacity of a system. The multiplexer creates a single analog voltage output to represent the state of up to four digital inputs. The DigitalInputDemux component then de-multiplexes the analog voltage into four StatusBoolean outputs.

Figure 112 DigitalInputDemux object application



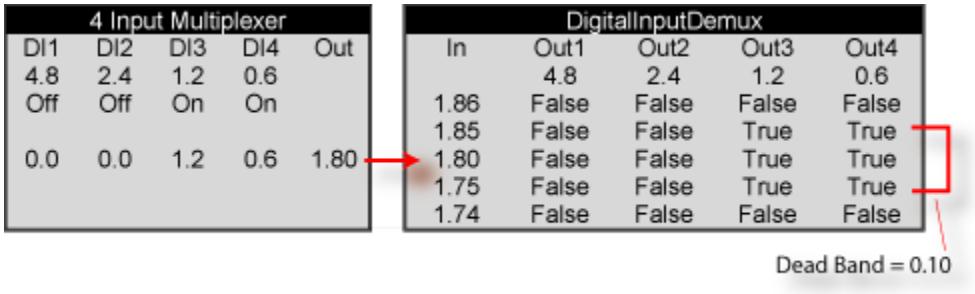
The illustration shows four volt-free contacts, such as door status, connected to a four-channel digital input multiplexer. The IO/DIM4 multiplexer device is manufactured by Titan Products Ltd., although similar suitable devices are available from other equipment suppliers. The IO/DIM4 multiplexer is microprocessor-based and designed to convert four separate digital input signals into a single analog voltage output. Each combination of digital input signal is converted to an output voltage level, which is then connected to one universal input of an NDIO IO-16 module. The NDIO universal input is configured as a voltage input point. Its output feeds the DigitalInputDemux object. Finally, the DigitalInputDemux object provides four status Boolean outputs, which represent the status of the original four contacts.

Property	Value	Description
In	number (defaults to 0.00) and null definition	Provides a StatusNumeric value for this object to serve as the input analog value from the multiplexer, which represents the state of the four digital inputs. This input must be valid for the object to function. When null is checked, the value displayed defaults to the incoming value from the device. If you remove the check mark you can configure the**In** value.
In Offset	positive or negative number to two decimal places	Defines an offset for the input.
Out1	read-only `true` or `false` (default)	Provides a status Boolean value, which the component sets to `true` if the **In** property contains a value equivalent to the value set as the **Out1 Value** property.
Out2	read-only `true` or `false` (default)	Provides a status Boolean value, which the component sets to `true` if the **In** property contains a value equivalent to the value set as the **Out2 Value** property.
Out3	read-only `true` or `false` (default)	Provides a status Boolean value, which the component sets to `true` if the **In** property contains a value equivalent to the value set as the **Out3 Value** property.
Out4	read-only `true` or `false` (default)	Provides a status Boolean value, which the component sets to `true` if the **In** property contains a value equivalent to the value set as the **Out4 Value** property.
Out1 Value	number to two decimal places (defaults to 4.80)	Sets a value, which corresponds with the equivalent setting in the multiplexer device to represent the status of the digital input 1.
Out2 Value	number to two decimal places (defaults to 2.40)	Sets a value, which corresponds with the equivalent setting in the multiplexer device to represent the status of the digital input 2.
Out3 Value	number to two decimal places (defaults to 1.20)	Sets a value, which corresponds with the equivalent setting in the multiplexer device to represent the status of the digital input 3.
Out4 Value	number to two decimal places (defaults to 0.60)	Sets a value, which corresponds with the equivalent setting in the multiplexer device to represent the status of the digital input 4.
Dead Band	number to two decimal places (defaults to 0.10)	Sets a tolerance value to prevent output 'chatter' due to a fluctuating input value. This property operates purely on the input **In** value.

Dead band example

An on-site multiplexer device with four digital inputs (DI) feeds the DigitalInputDemux component. DI1 and DI2 are open (off) and DI3 and DI4 are closed (on). The combined voltage weighting of DI3 and DI4 is 1.8v. An NDIO universal input transmits the combined voltage weighting of DI3 and DI4 (1.8v) to the DigitalInputDemux component. The DigitalInputDemux object then faithfully de-multiplexes this signal, setting **Out3** and **Out4** to `true`. In practice, a voltage drop occurs on the received signal and the **Dead Band** property allows you to engineer in some protection for this fluctuation to prevent it from adversely upsetting your control strategy.

Figure 113 Operation of the Dead Band property



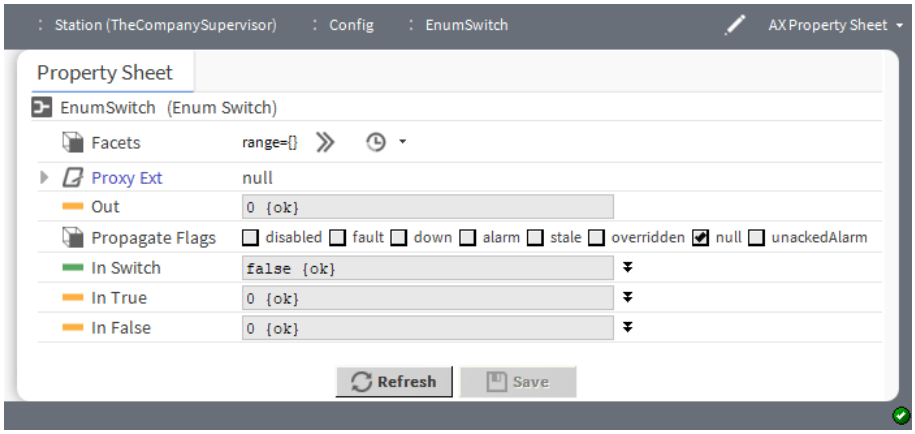
The **Dead Band** property in this example is set to 0.10, which is applied to the **In** value. The dead band function operates equally in both positive and negative sense on the **In** value. In this example therefore, all values from 1.75 through to 1.85 are valid.

kitControl-EnumSwitch

This component selects one of two StatusEnum inputs based upon the Boolean value at the StatusBoolean input **In Switch**.

EnumSwitch is available in the **Util** folder of the **kitControl** palette.

Figure 114 EnumSwitch properties



Property	Value	Description
Facets	Config Facets window (defaults to blank)	Defines a range of enumerated states (enum values).
Out	read-only StatusBoolean `true` or `false` (default)	Displays the result of the selection.
In Switch	`true` or `false` (default) and null definition	Indicates which input value to select.

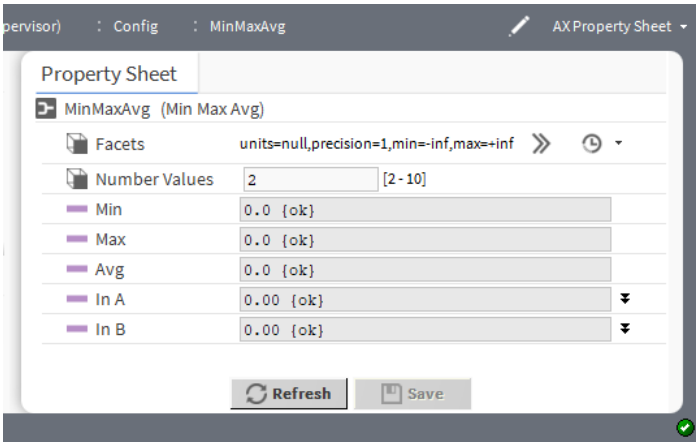
Property	Value	Description
In True	`true` or `false` (default) and null definition	Returns `true`.
In False	`true` or `false` (default) and null definition	Returns `false`.

kitControl-MinMaxAvg

This component has three StatusNumeric output slots that provide the current minimum, maximum, and average values of from 2 to 10 linked StatusNumeric inputs.

It is available in the **Util** folder of the **`kitControl`** palette.

Figure 115 MinMaxAvg properties



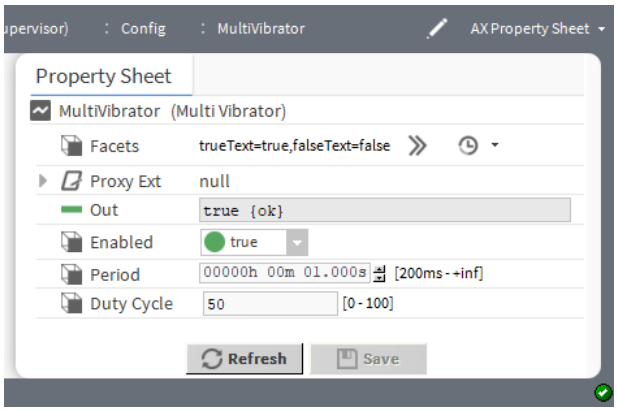
Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum= positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** value for this component.
Number Values	number (defaults to 2–10)	Select the number of inputs which need to be calculated.
Min	read-only number	Reports the lowest number in the set.
Max	read-only number	Reports the highest number in the set.
Avg	read-only number	Reports the average.
In A through In J	number (defaults to 0) and null definition	Defines the set of number upon which to calculate the average.

kitControl-MultiVibrator

This component provides an oscillating binary pulse output (StatusBoolean) with a period to configure between 200ms to infinity and a duty cycle from 0 to 100%.

MultiVibrator is available in the **Util** folder of the **kitControl** palette.

Figure 116 MultiVibrator properties



Property	Value	Description
Facets	Config Facets window (defaults to `true` and `false`)	Defines the text to display for each Boolean value. `trueText` configures the text to describe the state when the component returns `true`. `falseText` configures the text to describe the state when the component returns `false`.
Out	read-only StatusBoolean `true` (default) or `false`	Reports the result of the calculation.
Period	hours minute seconds (defaults to one minute)	Defines (from 200ms to infinity) how long it takes for an oscillation to complete an on-off cycle.
Duty Cycle	percentage	Defines the percentage of the **Period** in which the oscillation signal is active. This is the total period of the signal. A 40% duty cycle means the signal is on 40% of the time and off 60% of the time.

kitControl-NumericBitAnd

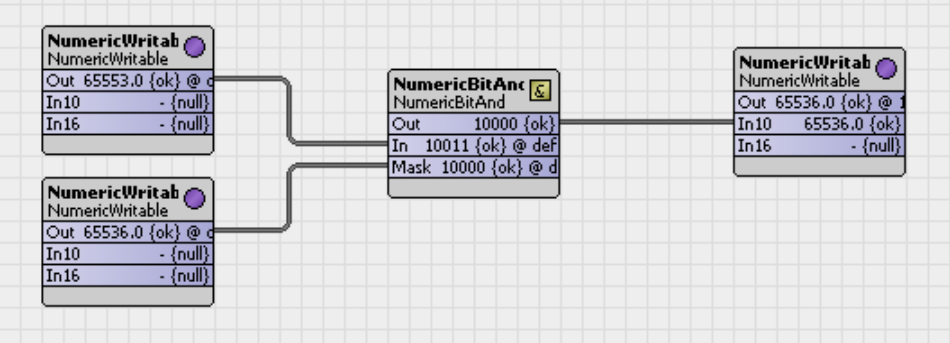
This component performs a logical AND on the bit equivalent of the StatusNumeric **In** value against the bit equivalent of its StatusNumeric mask slot value. It may be useful in cases where Boolean information is mapped into integer values.

NumericBitAnd is available in the **Util** folder of the **kitControl** palette, along with the closely-related NumericBitOr and NumericBitXor components.

As an example, some manufacturers multiplex binary data into a single numerical point by converting the bits from hexadecimal to decimal format. To obtain the status of the individual binary data, the number must be converted back from decimal to hex format. Each digit of the hex number represents a particular binary state (0 = false, 1 = true). The NumericBitAnd object converts a StatusNumeric input to its hex value and

compares it against the mask value. Any digits with a value of 1 in the mask and the input result in a corresponding value of 1 in the same digit of the output.

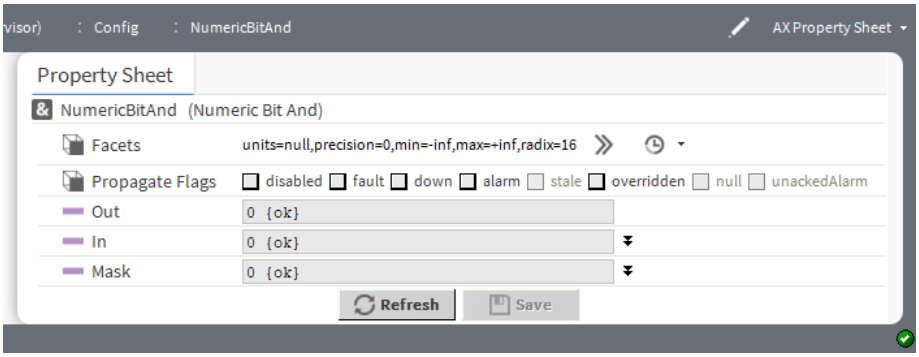
Figure 117 NumericBitAnd example



In this example:

- Input decimal 65553 converts to a hex value of 10011.
- Mask decimal 65536 converts to a hex value of 10000.
- The resulting hex value is 10000.

Figure 118 NumericBitAnd properties



Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum=positive infinity, and radix=16)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. `radix` defines the base of the numeration system. Radix 16 is hexidecimal. This definition applies to the **`Out`** property value for this component.
Out	read-only number (defaults to zero (0))	Displays the result of the comparison.

Property	Value	Description
In	number (defaults to zero (0)) and null	Defines the bit to compare with the **Mask**.
Mask	number (defaults to zero (0)) and null	Defines what to compare with the **In** property.

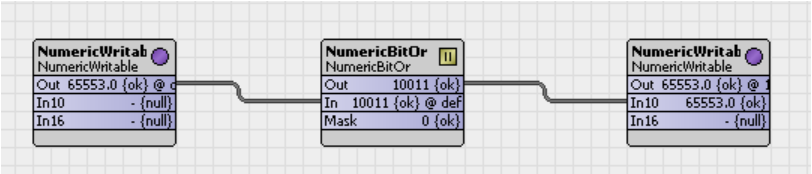
kitControl-NumericBitOr

This component performs a logical OR on the bit equivalent of the StatusNumeric **In** value against the bit equivalent of its StatusNumeric mask slot value. It may be useful in cases where boolean information is mapped into integer values.

NumericBitOr is available in the **Util** folder of the **kitControl** palette, along with the closely-related NumericBitAnd and NumericBitXor.

As an example, some manufacturers multiplex binary data into a single numerical point by converting the bits from hexadecimal to decimal format. To obtain the status of the individual binary data, the number must be converted back from decimal to hex format. Each digit of the hex number represents a particular binary state (0 = false, 1 = true). The NumericBitOr object converts a StatusNumeric input to a hex value, and compares it against the mask value. Any digits with a value of 1 in the mask or the input results in a corresponding value of 1 in the same digit of the output. Any value on the output slot greater than 1 indicates that at least one of the binary parameters is true.

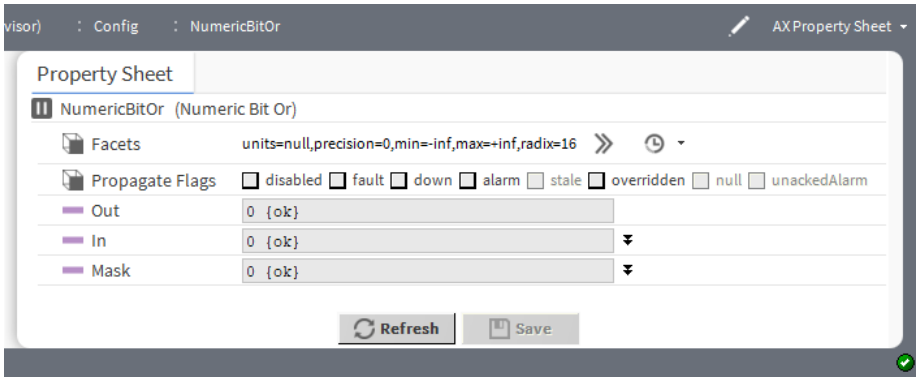
Figure 119 NumericBitOr example



In this example:

- Input decimal 65553 converts to a hex value of 10011.
- Mask decimal 0 converts to a hex value of 00000.
- The resulting hex value is 10011.

Figure 120 NumericBitOr properties



Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum=positive infinity, and radix=16)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. `radix` defines the base of the numeration system. Radix 16 is hexidecimal. This definition applies to the **`Out`** property value for this component.
Out	read-only number (defaults to zero (0))	Displays the result of the comparison.
In	number (defaults to zero (0)) and null definition	Defines the bit to compare with the **`Mask`**.
Mask	number (defaults to zero (0)) and null definition	Defines what to compare with the **`In`** property.

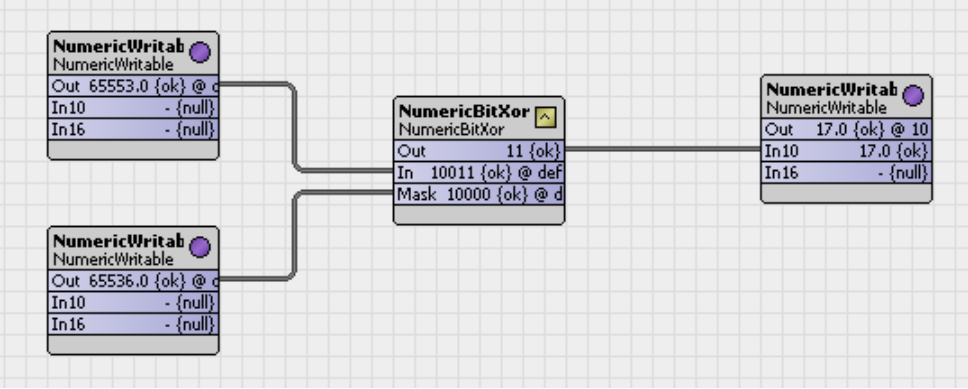
kitControl-NumericBitXor

This component performs a logical XOR on the bit equivalent of the StatusNumeric **`In`** value against the bit equivalent of its StatusNumeric mask slot value. It may be useful in cases where boolean information is mapped into integer values.

NumericBitXor is available in the **Util** folder of the **`kitControl`** palette, along with the closely-related NumericBitAnd and NumericBitOr.

As an example, some manufacturers multiplex binary data into a single numerical point by converting the bits from hexadecimal to decimal format. To obtain the status of the individual binary data, the number must be converted back from decimal to hex format. Each digit of the hex number represents a particular binary state (0 = false, 1 = true). The NumericBitXor object converts a StatusNumeric input to hex value and compares it against the mask value. Each digit is analyzed using exclusive OR (XOR) logic, setting the corresponding digit value to either a 1 or 0.

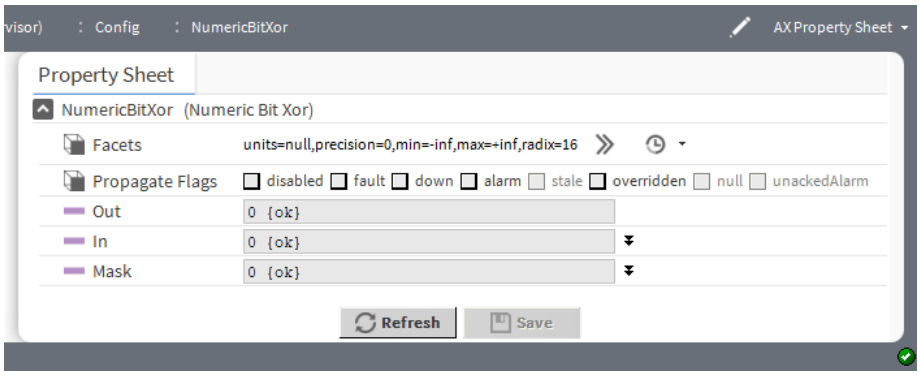
Figure 121 NumericBitXor example



In this example:

- Input decimal 65553 converts to a hex value of 10011.
- Mask decimal 65536 converts to a hex value of 10000.
- The resulting hex value is 00011.

Figure 122 NumericBitOr properties



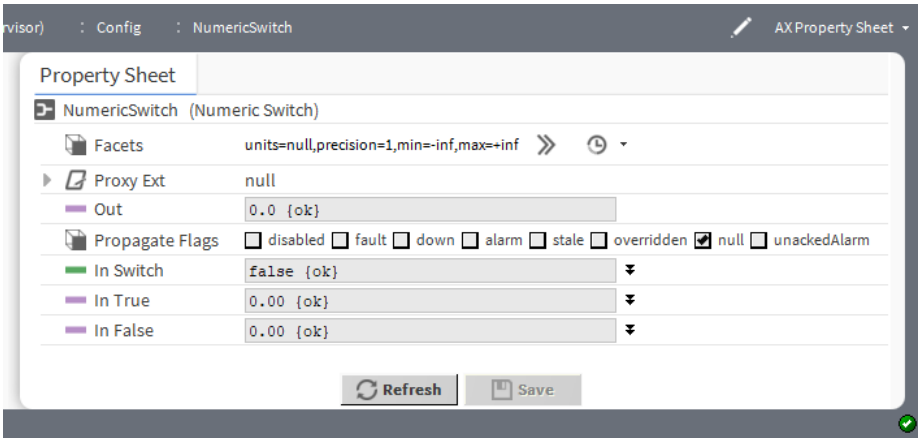
Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, maximum=positive infinity, and radix=16)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. `radix` defines the base of the numeration system. Radix 16 is hexidecimal. This definition applies to the **`Out`** property value for this component.
Out	read-only number (defaults to zero (0))	Displays the result of the comparison.
In	number (defaults to zero (0)) and null definition	Defines the bit to compare with the **`Mask`**.
Mask	number (defaults to zero (0)) and null definition	Defines what to compare with the **`In`** property.

kitControl-NumericSwitch

This component selects one of two StatusNumeric inputs based upon the Boolean value at the StatusBoolean input **`In Switch`**.

NumericSwitch is available in the **Util** folder of the **`kitControl`** palette.

Figure 123 BooleanSwitch properties



Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, and maximum=positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** property value for this component.
Out	number (defaults to zero (0)) and null definition	Displays the result of the selection.
In Switch	`true` or `false` (default) and null definition	Indicates which input value to select.
In True	`true` or `false` (default) and null definition	Returns `true`.
In False	`true` or `false` (default) and null definition	Returns `false`.

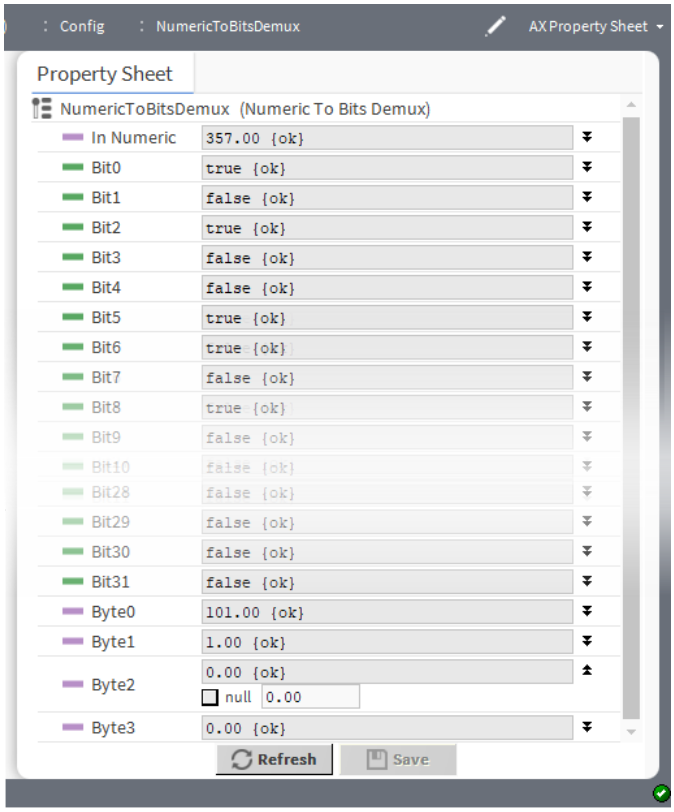
kitControl-NumericToBitsDemux

This component converts a numeric value into the binary equivalent. Each bit in the component represents the binary bit position of the numeric integer (the component truncates numerics to whole numbers for the conversion). This component can express numeric values in bits (up to 32) as well as bytes (up to 4).

NOTE: This component is not designed to convert negative numbers.

NumericToBitsDemux component is located in the **Util** folder of the **`kitControl`** palette.

Figure 124 NumericToBitsDemux properties



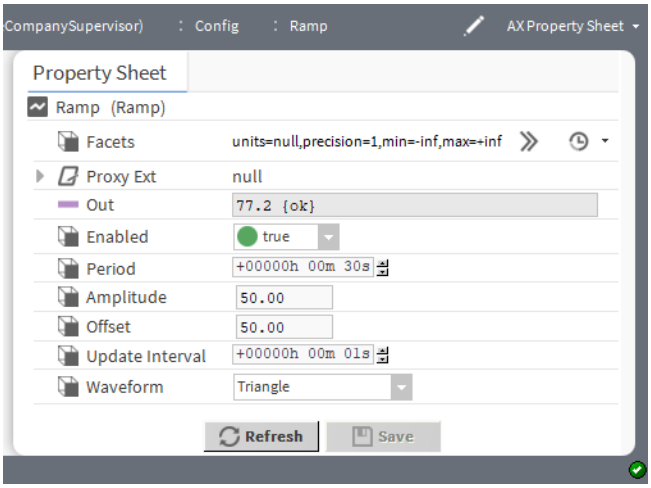
Property	Value	Description
In Numeric	number (defaults to 0.00) and null definition	Displays the numeric input value. Typically, you would link a StatusNumeric output to the **`In Numeric`** property of this component. The component propagates the Status portion of the input to all StatusBoolean and StatusNumeric (byte) outputs. When null is checked, the corresponding value defines the default when a calculation returns null.
Bit0 through Bit31	`true` or `false` (default) and null definition	Represent the converted numeric as binaries. When null is checked, the corresponding value defines the default when a calculation returns null.
Byte0 through Byte3	number (defaults to 0.00) and null definition	Express the converted numeric input as bytes. When null is checked, the corresponding value defines the default when a calculation returns null.

kitControl-Ramp

This component provides a StatusNumeric Out with a linear ramping output. Properties define the Period, Amplitude, Offset, and Update Interval.

Ramp is available in the **Util** folder of the **`kitControl`** palette.

Figure 125 Ramp properties



Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, and maximum=positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** property value for this component.
Out	read-only number	Reports the result of the calculation.
Period	hours minutes seconds (defaults to 30 seconds)	Defines the duration of the ramp.
Amplitude	number of decibels (defaults to 50.00)	Amplitude is height of the ramp from its lowest to highest point.
Offset	positive or negative number to two decimal places (defaults to 50.00)	Adjusts the amplitude.
Update Interval	hours minutes seconds (defaults to one second)	Specifies the time between output changes. A smaller value results in a more accurate ramp with more changes per second, while a larger value results in less precision but with less overhead.

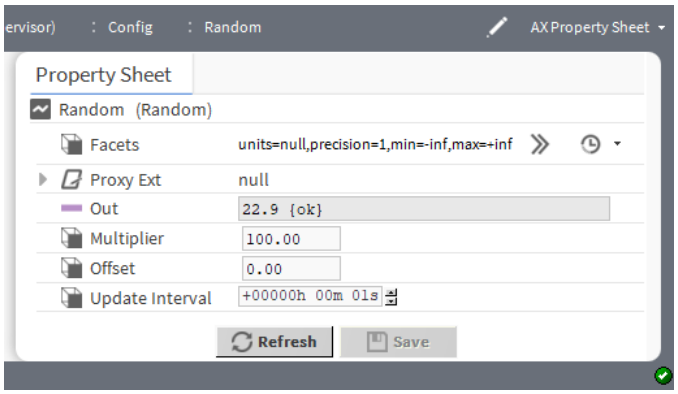
Waveform	drop-down list	Selects the type of waveform:
		`Triangle`
		`Saw Tooth`
		`Inverted Saw Tooth`

kitControl-Random

This component generates random numbers. The output is derived by multiplying a random number (that is greater than 0 but less than 1) times a variable multiplier plus an offset. See the next section, Random setup.

Random is available in the Util folder of the **kitControl** palette.

Figure 126 Random properties



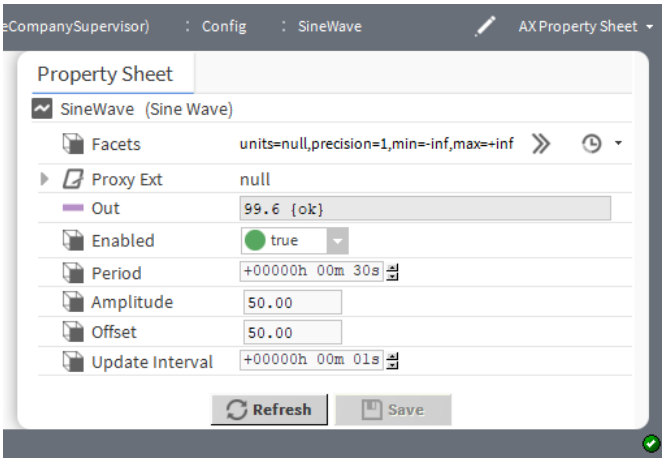
Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, and maximum=positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **Out** property value for this component.
Out	read-only number to a single decimal place	Displays the random number.
Multiplier	number to two decimal places (defaults to 1.0)	Defines a double value used to multiply by the random number (the random number is: `>=0.0 but <1.0`).
Offset	number to two decimal places (defaults to 50)	Defines a positive or negative distance from zero that the wave's amplitude is centered on.
Update Interval	hours minutes seconds (defaults to one second)	Defines the amount of time between output changes.

kitControl-SineWave

This component generates a sine wave as a StatusNumeric out.

SineWave is available in the **Util** folder of the **kitControl** palette.

Figure 127 SineWave properties



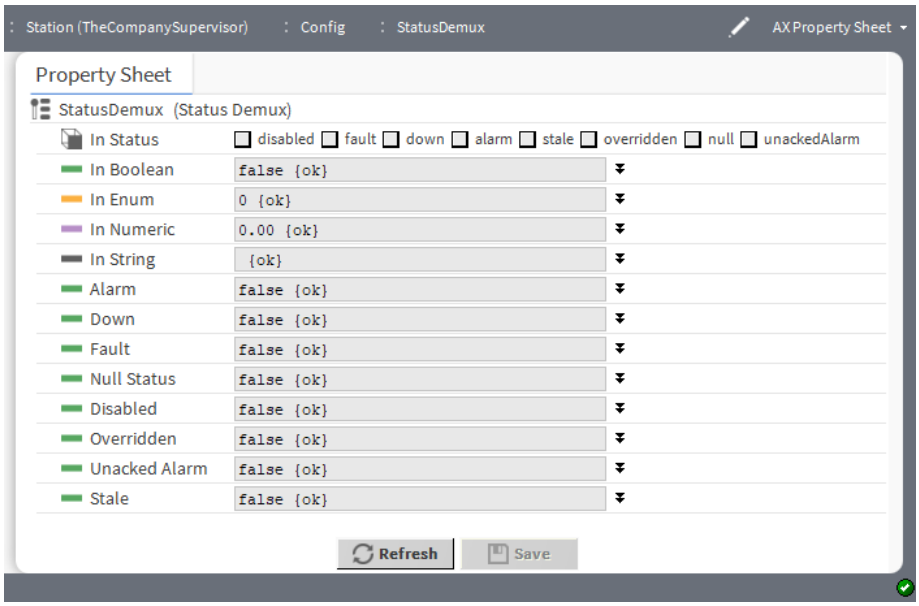
Property	Value	Description
Facets	Config Facets window (defaults to null units, a single decimal place, minimum=negative infinity, and maximum=positive infinity)	Selects units and configures how the value displays: `units` defines the unit of measure from`acceleration` to `volumetric flow`. `precision` defines the number of decimal places. `min` defines the lowest value allowed. `max` defines the highest value allowed. This definition applies to the **`Out`** property value for this component.
Out	read-only number	Displays the calculated value
Period	hours minutes seconds (defaults to 30 seconds)	Defines the duration of the ramp.
Amplitude	number of decibels (defaults to 50.00)	Amplitude is the height of sine wave from its lowest to highest point.
Offset	positive or negative number to two decimal places (defaults to 50.00)	Adjusts the amplitude.
Update Interval	hours minutes seconds (defaults to one second)	Defines how often to restart the ramp.
Offset	number to two decimal places (defaults to 50)	Defines a positive or negative distance from zero that the wave's amplitude is centered on.
Update Interval	hours minutes seconds (defaults to one second)	Defines the interval.

kitControl-StatusDemux

This component provides a method to check for individual status flags of the In-linked object, and sets corresponding (demuxed) StatusBoolean out slots active (true) if that status as found.

StatusDemux is available in the **Util** folder of the **kitControl** palette.

Figure 128 StatusDemux properties



Property	Value	Description
In Status	check boxes	Identifies the possible conditions of the In-linked object: disabled, fault, down, alarm, stale, overridden, null and unackedAlarm.
In Boolean	`true` or `false` (default) and null definition	Receives input from a connected Boolean object.
In Enum	enumerated value (defaults to zero) and null definition	Receives input from a connected Enum object.
In Numeric	number (defaults to zero) and null definition	Receives input from a connected Numeric object.
In String	text (defaults to blank) and null definition	Receives input from a connected String object.
Alarm	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object is in alarm.
Down	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object is down.
Fault	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object is in fault.

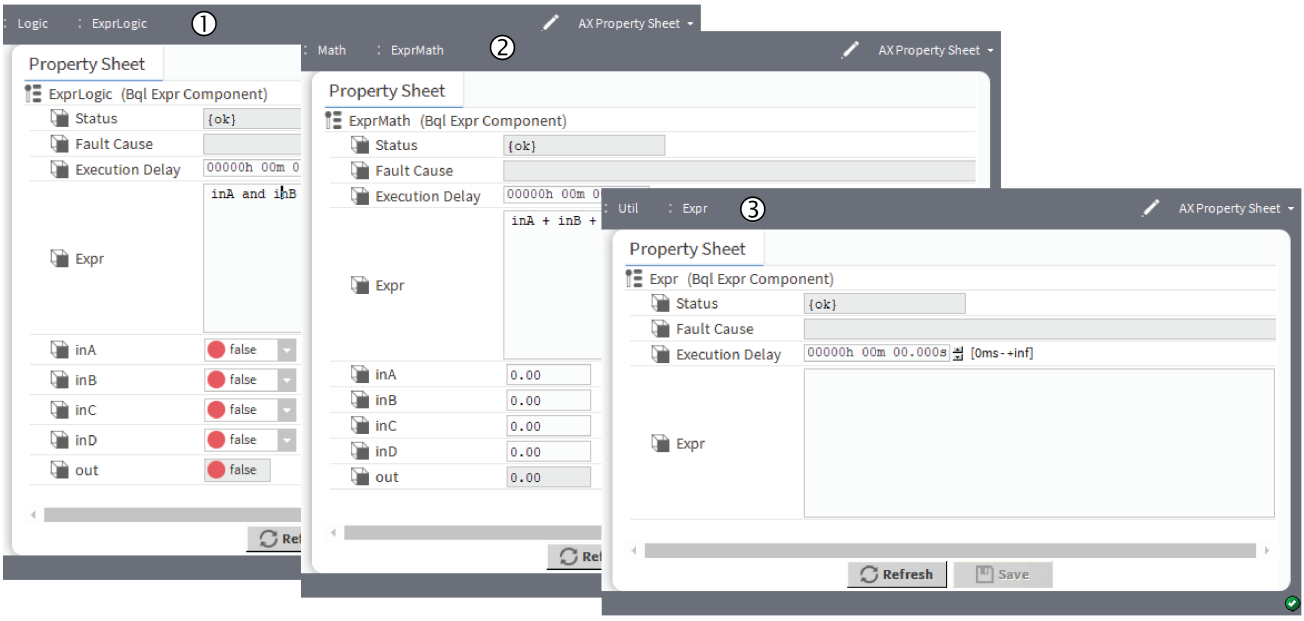
Property	Value	Description
Null Status	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object is in null status.
Disabled	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object reports a disabled.
Overridden	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object has been overridden.
Unacked Alarm	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object has an unacknowledged alarm
Stale	out property: `true` or `false` (default) and null definition	Activates (`true`) if the **In**-linked object is stale (has not been updated in a timely manner).

kitControl-BqlExprComponent

This component creates custom math and logic operations based upon manually-added slots and one or more BQL expression statements. Slots can be various baja types, such as primitives Double, Float, Integer, Boolean, or String, or status types: StatusBoolean, StatusNumeric, and so on. Slots are either inputs or one or more outputs. You enter BQL expressions in the component's **Expr** (expression) property.

The ExprLogic and ExprMath components are not strictly speaking logic and math components. Instead, they are adaptations of the Expr component (BqlExprComponent), which provide four-input logic and math functions.

Figure 129 The three forms of the BqlExprComponent



- (1) is for logic BQL expressions.
- (2) is for math BQL expressions.
- (3) is for general expressions.

The **Logic** and **Math** folders also contain expression components. ExprLogic provides a four-input logic AND gate. ExprMath provides a four-input math ADD component. The **Util** folder of the **kitControl** palette contains an empty expression component.

For complete information on BQL expressions, refer to Niagara Engineering Notes,*BQL Expression component*.

Property	Value	Description
Status	read only	Indicates the condition of the component at the last check {ok} indicates that the component is licensed and polling successfully. {down} indicates that the last check was unsucessful, perhaps because of an incorrect property, or possibly loss of network connection. {disabled} indicates that the **Enable** property is set to false. false indicates another problem . Refer to **Fault Cause** for more information.
Fault Cause	read only	Indicates a reason why a system object (network, device, component, extension, etc.) is in fault. This property is empty unless a fault exists.
Execution Delay	hours minutes seconds (defaults to 0)	Configures an amount of time to delay before executing the expression.
Expr	BQL expression	The standard syntax for an expression is as follows: input operator 'output', where: input is the name of one or more slots. operator is a word or symbol. 'output' is the slot that contains the result of the expression. (note apostrophes around slot name)
InA, In B, InC and InD	BooleanStatus (true or false for logic), number (for math)	Defines four input values.
out	read-only BooleanStatus (true or false for logic), number (for math)	Defines a single output value, which is the result of executing the expression.

The supported operators for the BQL expression are as follows:

Order of Operation	Operator	Description
1	!, not, -	Logical not, numeric negation
2	* , /	multiplication, division
3	+, -	addition, subtraction
4	=, !=, >, >=, < <=, like	Comparison

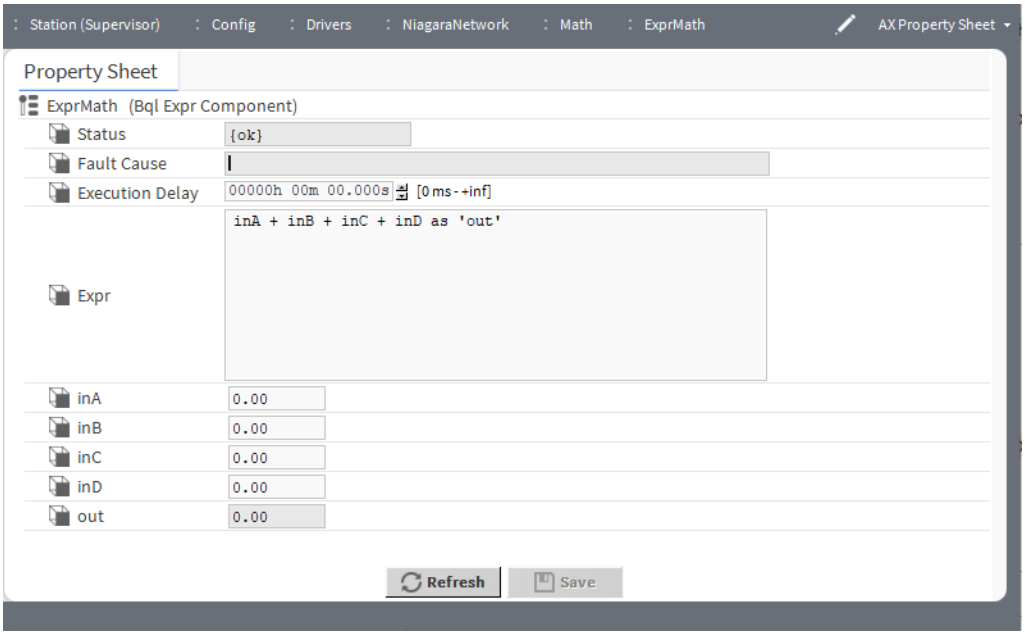
5	and, or	Logical operators
6	as	result operator

Operators are processed by their precedence, that is “order of operation”, from first (1) to last (6). Parentheses is used to override the normal precedence.

Comma is used as a delimiter to create **`Expr`** with multiple expression(output slots).

For long statements in a **`Expr`** use CR/LF to wrap the text.

BQL Like Query



Following are some examples for Like expression:

- Multiply four Double properties
  inA * inB * inC * inD as ‘out’
- Multiply two BStatusNumeric properties to a Double output property
  inA.value * inB.value as ‘out’
- Negate a Double input property (if inA is 5, out becomes -5)
  -inA as ‘out’