# Sedona Framework

For those who understand Tridium's Niagara Framework, understanding Sedona Framework would be trivial. However, the market for Sedona Framework goes beyond the traditional HVAC market that Niagara Framework serves. Sedona Framework is intended to provide a complete development environment for devices with limited resources. The Java environment of Niagara Framework requires resources well beyond the capability of small devices but much of the Niagara concepts have been transferred down to Sedona Framework thereby allowing small devices to execute sophisticated control.

## Sedona Workbench

In order to understand how Sedona is being used, we need to define some terms commonly used with either of the two Tridium Framework products. Sedona Workbench is PC-based graphical user interface (GUI) development tool that is used to develop Sedona applications. This connects to the ISMA Sedona Controllers (the target platform) over IP/Ethernet. The Sedona Framework is comprised of kits that contain components; whereas components are used to implement Sedona applications. Components have properties that can be shown on Property Sheets and can be interconnected on Wire Sheets using virtual wires. The user opens up a SOX (Sedona protocol) session on Workbench to the controller. Using drag-and-drop methodology, the user develops a control scheme by interconnecting components and by assigning properties to components. The final application is stored on the controller and saved on Workbench. A Sedona Virtual Machine (SVM), which permanently resides in the controller, executes the Sedona code sent over by Workbench. However, as the application executes, the user can observe the execution on Workbench. With Workbench disconnected, the application continues to run since the application is stored in flash memory on the controller.

## Palette, Kits, and Components

Sedona Framework provides a Palette of kits, one of the more interesting kits is called iSMA_control which includes a group of components similar to the components found in Niagara Framework. Although not as numerous, the Sedona control components are still very powerful. There are three types of variables that can be operated upon – floating point (float), binary (Boolean), and Integer (Int). For better understanding, the components in the control kit are shown sub-divided into functional groups.

## Conversion Components

Contains 8 components that mainly convert from one point type to another.

**BooleanToFloat** »

BooleanToFloat is a 16 bit Binary to Float Encoder Object.

out := Encoded value of inputs, with bit15(MSB) and bit0(LSB)
count := Sum of the inputs that are active

**BooleanToPulse** »

BooleanToPulse is a simple mono-stable oscillator object.

out := in for one scan cycle on the rising edge of in

**FloatToBoolean** »

FloatToBoolean is a 16-bit Float to Binary Decoder Object.

Outputs (bit15-bit0) := Decoded value of input, with bit15(MSB) and bit0(LSB)
overflow := true when inNumeric > 65535

**FloatToInteger** »
FloatToInteger is a Float to 32 bit integer (Integer) Converter Object.

out := in, except that the output is the 32-bit integer with fractional part truncated.

**FloatToLong** »
FloatToLong is a Float to 64-bit signed integer (Long) Converter Object.

out := in, except that the output is the 64-bit signed integer with fractional part truncated.

**FloatToString** »
FloatToString is a Float to String (Buf) Converter Object.

out := in, except that the output is the Buf(64) with fractional part truncated.

**IntegerToFloat** »
IntegerToFloat is a 32 bit integer (Integer) to Float Converter Object.

out := in, except that the output is the float.

**LongToFloat** »
LongToFloat is a 64-bit signed integer (Long) to Float Converter Object.

out := in, except that the output is the float.

## Control extensions

As needed, you can add control extensions to points along with alarm and history extensions. Control extensions perform additional processing on a point's received value. There are relatively few types of control extensions.

**DiscreteTotalizer** 
DiscreteTotalizer accumulates runtime and change of state (COS) count. Extension actions permit resetting (zeroing) the runtime and COS count.

**IntegerTotalizer** 
IntegerTotalizer accumulates integer total using hourly or minutely totalization. Extension has action to reset (zero) total.

**NumericTotalizer** 
NumericTotalizer accumulates numeric total using hourly or minutely totalization. Extension has action to reset (zero) total.

For example, a totalizer with a minutely totalization interval can convert an instantaneous flow reading in cubic feet per minute (cfm) into a value representing total cubic feet consumed.

## Demux components

Demux objects selects one of two outputs to receive the Input value, depending on the value of Boolean select Input. The value of the other Output remains unchanged.

**BooleanDemux** 
BooleanDemux object selects one of two outputs to receive the Input (Boolean) Value, depending on the value of the Boolean select Input. The value of the other Output remains unchanged.

| select | out1 | out2 |
|---|---|---|
| false | in | Previous-Value |
| true | Previous-Value | in |

**IntegerDemux** 

IntegerDemux object selects one of two outputs to receive the Input (Integer) Value, depending on the value of the Boolean select Input. The value of the other Output remains unchanged.

| select | out1 | out2 |
|---|---|---|
| false | in | Previous-Value |
| true | Previous-Value | in |

**NumericDemux** 

NumericDemux object selects one of two outputs to receive the Input (Numeric) Value, depending on the value of the Boolean select Input. The value of the other Output remains unchanged.

| select | out1 | out2 |
|---|---|---|
| false | in | Previous-Value |
| true | Previous-Value | in |

## Energy components

Energy 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.

**DegreeDays** 

DegreeDays provides degree day calculations, based upon temperature received at the input Temperature slot and values of various other properties.

Definition of Degree Days: 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.

The DegreeDays component includes the following properties and one action:

**Unit Select** : This is used to set the units of the Temp In, Min Temp, Max Temp, and Mean Temp properties.
**Base Temperature** : Specifies the base temperature used in the degree-day calculation.
**Input Temperature** : This is the input for the outside air temperature used in the degree-day calculation. Note: If this input is not valid then no calculations will be done.
**Minimum Temperature** : The minimum temperature recorded for the current day. Tested and set on each calculation.
**Maximum Temperature** : The maximum temperature recorded for the current day. Tested and set on each calculation.
**Mean Temperature** : The mean temperature recorded for the previous day. Calculated when the day changes. Mean Temp = (Max Temp + Min Temp) / 2.0
**Cooling Degree-day** : This is the cooling degree-day calculated for the previous day. Calculated when the day changes.
**Totalized Cooling Degree-days** : This is the totalized cooling degree-days since last Reset Totals action was invoked. Calculated when Cooling Degree Days changes.
**Heating Degree-day** : This is the heating degree-day calculated for the previous day. Calculated when the day changes.
**Totalized Heating Degree-days** : This is the totalized heating degree-days since last Reset Totals action was invoked. Calculated when Heating Degree Days changes.

**NightPurge** 

This component is available in the iSMA_control palette. It uses the two sets of temperature and humidity inputs to find the air supply with the least amount of heat when the purgeEnabled input is true. The freeCooling output will be set to false if outside >= inside or set to true if outside = nightSetpoint.
For inside and outside comparisons, you can select either temperature or enthalpy comparisons. There is also a low temperature check to protect against freezing.

The NightPurge component includes the following properties:

**Unit Select** : Specifies the units of Temperature and Humidity properties.
**Purge Enabled** : Boolean, must be true to enable night purge operation. Whenever false, the Free Cooling output is set 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, Purge Enabled is linked to a "Not" object sourced from a BooleanSchedule output.
**Outside Temperature** : Input for the current outside air temperature. This input must be valid for this object to function.
**Outside Humidity** : Input for the current outside air humidity. This input must be valid for this object to function.
**Inside Temperature** : Input for the current inside air temperature. This input must be valid for this object to function.
**Inside Humidity** : Input for the current inside air humidity. This input must be valid for this object to function.
**Low Temperature Limit** : This property is used to provide freeze protection.
**Night Setpoint** : Inside night temperature setpoint, at or below which free cooling is not applied. Instead, the Current Mode is set to "Satisfied."
**Outside Enthalpy** : This is the calculated outside air enthalpy.
**Inside Enthalpy** : This is the calculated inside air enthalpy.
**Free Cooling** : A Boolean output set to value of the Free Cooling Command when it is determined that free cooling should be used. Otherwise, the value is set to the opposite state, or null (if Used Null Output is set to true).
**Current Mode** : This enumeration indicates which of the following modes this object is currently in:

- Disabled (Purge Enabled is false)
- Free Cooling
- No Free Cooling (free cooling not available)
- Low temperature (Outside Temp below Low Temperature Limit, free cooling disabled)
- Input error (A temperature or humidity is invalid (down, fault, etc.), free cooling disabled)
- Satisfied (Inside temperature below Night Setpoint, free cooling disabled)

**Setpoint Deadband** : Temperature setpoint deadband applied when inside temperature falls below Night Setpoint, before free cooling can be enabled. Default value is 1.0.
**Threshold Span** : The difference between the inside enthalpy and the outside enthalpy must be greater than this value before free cooling will be enabled. Default value is 1.0.
**Use Enthalpy** : Setting this property to true will enable the use of enthalpy for determining if free cooling is available. Otherwise, it will just use outside and inside temperature to decide.

**OptimizedStartStop**

The OptimizedStartStop component allows you to use Start Time Optimization and Stop Time Optimization to save energy. This component 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.

The two basic optimization types are described, as follows:

**Start time optimization** : This type of optimization reduces energy consumption by turning on equipment at the latest possible time that still allows for providing a comfortable temperature by occupancy time.

**Stop time optimization** : This type of optimization turns equipment off at the earliest possible time that allows the building to "drift" and stay within a temperature comfort range until the end of occupancy time.

The OptimizedStartStop calculation is performed at 15 seconds after the beginning of every minute, when the appropriate Start Enable or Stop Enable properties are set to true, a valid schedule event is linked to the component, and the next scheduled event value is not already set.

For example, if a value is scheduled to be set to "true" in 1 hour but is already set to "true", no calculation is performed, even if the Start Enable or Stop Enable properties are set to true. The product of this calculation is the "Calculated Command Time". The Calculated Command Time applies to both the Start Time and the Stop Time, as appropriate. Therefore, it defines an early start command to achieve a specified temperature range by occupancy time or an early stop command without sacrificing the temperature range by unoccupancy time. After a CalculatedCommand Time is invoked, the actual area response (temperature change rate) is analyzed and weighted adjustments are made to the calculation parameters based on the detected values so that subsequent calculations might be more accurate.

Start time and stop time operations are described below:

**Calculated Start Time** : Only one optimized start sequence is performed per day. The following factors affect the Calculated Start Time calculation.

**Temperature differential** : If the 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** : The run-time heating or cooling factors (depending on the direction the space temperature must move) are multiplied 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** : When the system's time is later than the schedule's time offset by the calculated leadtime, the optimum start outputs are enabled.

**If the calculated leadtime is so large that an optimum start time prior to midnight is the result, the optimum start occurs at midnight. An optimum start is performed only for the first scheduled start for the day.

**Calculated Stop Time** : You can perform multiple stop operations but no optimized stop can occur before the time specified by the Earliest Stop Time property.

**Temperature differential** : If the space temperature is inside the range defined by the lower and upper comfort limits and 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** : Optimum stop time is invoked for each of the schedule's inactive events and is based on the drift time and Next Event Time value.

The OptimizedStartStop component includes the following properties:

**Heat Cool Mode** : This boolean property allows you to enable either the heatMode or the coolMode. The selected option applies only to optimized stop calculations which 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.

**Parameter Reset Time** : This property displays the time when any of the four runtime or drifttime properties change to the User Defined values. The OSS component copies the user defined drifttime and runtime property values to the corresponding actual drifttime and runtime property values.

**Start Enable** : This property allows you to manually or automatically enable or disable the optimized start function.

**Stop Enable** : This property allows you to manually or automatically enable or disable the optimized stop function.

**Schedule Status** : This boolean property monitors and displays the status of the schedule that is linked to it.

**Next Event Time** : This property is linked to a schedule for the time of the next scheduled event.

**Next Event Value** : This property is linked to a schedule and reflects the value of the action for next scheduled event.

**Outside Temp** : This property is linked to outside temperature and displays the value for information only.

**Space Temp** : This property is linked to a space temperature output and displays the temperature of the area affected by equipment associated with the OSS component.

**Start Time Command** : This boolean property is an output that you link to a control for invoking an equipment start command. For example, it can be linked to a prioritized input of a boolean writable - or directly to the equipment Start control.

**Stop Time Command** : This boolean property is an output that you link to a control for invoking an equipment stop command. For example, it can be linked to a prioritized input of a boolean writable - or directly to the equipment Stop control.

**Upper Comfort Limit** : This property value is the Cooling mode target temperature.

**Lower Comfort Limit** : This property value is the Heating mode target temperature.

**Dynamic Parameter Adjust** : This controls whether or not calculation parameters are programmatically adjusted after an execution. After the OSS component completes a start or stop control, if this property value is set to true, the component evaluates the actual recovery rate (degrees/hour) and automatically adjusts the Runtime and Drifttime properties values so that they are influenced by actual drift time and run time.

**Old Parameter Multiplier** : This property is used to weight the dynamic parameter adjustment calculation. The value that you specify in this field affects how much weighting you assign to the previous runtime property value when it is used in the dynamic parameter adjustment calculation. A larger value increases the amount of weighting given to the previous runtime and a smaller value decreases the weighting.

**Earliest Start Time** : This property allows you to specify a time, before which, no optimized start command may be issued. If this value is set earlier than the Calculated Command Time, the Calculated Command Time is adjusted to equal this time.

**Earliest Stop Time** : This property allows you to specify a time, before which, no stop command may be issued. If this value is set earlier than the Calculated Command Time, the Calculated Command Time is adjusted to equal this time.

**Drifttime Per Degree Cooling User Defined** : This property allows you to set a default value for calculating the rate of drift in cooling mode. When you save a value to this field, the value is copied to the Drifttime Per Degree Cooling field.

**Drifttime Per Degree Heating User Defined** : This property allows you to set a default value for calculating the rate of drift in heating mode. When you save a value to this field, the value is copied to the Drifttime Per Degree Heating field.

**Runtime Per Degree Cooling User Defined** : This property allows you to set a default value for calculating the runtime value in cooling mode. When you save a value to this field, the value is copied to the Runtime Per Degree Cooling field.
**Runtime Per Degree Heating User Defined** : This property allows you to set a default value for calculating the runtime value in heating mode. When you save a value to this field, the value is copied to the Runtime Per Degree Heating field.
**Drifttime Per Degree Cooling** : This property displays the actual value that is used for calculating an optimized stop time when the equipment is in cooling mode. This value is adjusted automatically if the Dynamic Parameter Adjust value is set to true.
**Drifttime Per Degree Heating** : This property displays the actual value that is used for calculating an optimized stop time when the equipment is in heating mode. This value is adjusted automatically if the Dynamic Parameter Adjust value is set to true.
**Runtime Per Degree Cooling** : This property displays the actual value that is used for calculating an optimized start time when the equipment is in cooling mode. This value is adjusted automatically if the Dynamic Parameter Adjust value is set to true.
**Runtime Per Degree Heating** : This property displays the actual value that is used for calculating an optimized start time when the equipment is in heating mode. This value is adjusted automatically if the Dynamic Parameter Adjust value is set to true.
**Last Start Time** : This is a record of the last Start Time that was used for calculating an optimized start time. Since only one optimized start per day is allowed, this value does not display Start Times (restarts) that are subsequent to the initial Start Time for a day.
**Last Stop Time** : This is a record of the last Stop Time that was used for calculating an optimized stop time. Since multiple Optimized Stops are allowed in a day, this value changes to reflect the latest Optimized Stop time.
**Outside Temp At Beginning** : This is a record of what the outside air temperature was at the time of the last start or stop command. This is the temperature that was used in calculations for dynamic parameter adjustment.
**Space Temp At Beginning** : This is a record of what the space temperature was at the time of the last start or stop command. This is the temperature that was used in calculations for dynamic parameter adjustment.
**Calculated Command Time** : This field shows the calculated time for the next command. This could be a start or a stop command.
**Program Mode** : As part of the logic that the OSS component uses, there are five "program mode" states. These states serve primarily in logic control, however, they may be informative to the system engineer, as well. The Program Mode value displays the current heating or cooling state for optimized start or stop. The following list describes the possible display values and meanings.

**0** ("No" Calculation)
This value indicates that no calculation is being made

**1** ("Start" Calculation)
This valued indicates that the optimized start calculation process is ongoing but that an optimized start or stop is not yet in progress.

**2** ("Start" in Process)
This value indicates that an optimized start has been initiated.

**3** ("Stop" Calculation)
This value indicates that an optimized stop calculation process is ongoing but that an optimized start or stop is not yet in progress.

**4** ("Stop" in Process)
This value indicates that an optimized stop has been initiated.

**OutsideAirOptimization**

OutsideAirOptimization is available in the iSMA_control palette. The OutsideAirOptimization component is used to support applications that need to allow for enthalpy based free cooling. This object is typically used during occupancy periods.

The freeCooling output is set to false if outside >= inside and set to true if outside <= inside - (abs) thresholdSpan. You can select temperature or enthalpy comparisons. There is also a low temperature check to protect against freezing.

Setup of the object involves the following properties, as follows:

**Unit Select** : This is used to set the units of the Temperature and Humidity properties.

**Outside Temperature** : Input for the current outside air temperature. This input must be valid for this object to function.

**Outside Humidity** : Input for the current outside air humidity. This input must be valid for this object to function.

**Inside Temperature** : Input for the current inside air temperature. This input must be valid for this object to function.

**Inside Humidity** : Input for the current inside air humidity. This input must be valid for this object to function.

**Low Temperature Limit** : This property is used to provide freeze protection.

**Outside Enthalpy** : This is the calculated outside air enthalpy.

**Inside Enthalpy** : This is the calculated inside air enthalpy.

**Free Cooling** : This boolean output value is set to the value of the Free Cooling Command when it is determined that free cooling should be used. Otherwise, the value is set to null.

**Current Mode** : This indicates what mode this object is currently in.

- Input out of range
- Free Cooling
- No Free Cooling
- Low temperature
- Input error

**Threshold Span** : The difference between the inside enthalpy and the outside enthalpy must be greater than this value before free cooling will be enabled.

**Use Enthalpy** : Setting this property to true will enable the use of enthalpy for determining if free cooling is available. Otherwise, it will just use outside and inside temperature to decide.

**Psychrometric**

The Psychometric component is available in the iSMA_control palette. You can use it to support applications that need to calculate the properties of moist air using given temperature and humidity inputs.

Setup of the component involves setting the following properties:

**Unit Select** : Used to set the units of the Temperature and Humidity properties.

**Input Temperature** : Input temperature

**Input humidity** : Input humidity

**Dew Point Temperature** : Calculated dew point temperature. Requires valid Input Temperature and Input Humidity to calculate.

**Enthalpy** : Calculated enthalpy. Requires valid Input Temperature and Input Humidity to calculate.

**Saturated Pressure** : Calculated saturated pressure. Requires valid Input Temp to calculate.

**Vapor Pressure** : Calculated vapor pressure. Requires valid Input Temperature and Input Humidity to calculate.

**Wet Bulb Temperature** : Calculated wet bulb temperature. Requires valid Input Temperature and Input Humidity to calculate.

## HVAC components

HVAC components provide various control functions used in commercial HVAC applications. Included are the following components:

### LeadLagCycles 

LeadLagCycles provides lead-lag control of 2 to 16 loads 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 will be active at a time based on cycle count.
LeadLagCycles is available in the iSMA_control palette, along with a similar LeadLagRuntime object.
Setup of the object involves the following properties:

**In** : A Boolean input that controls whether any control device should be on. If this input is true, one of the outputs will be active based on the cycle count of each controlled device.
**Number Outputs** : Specifies the number of devices (outputs) that are controlled.
**Max Runtime** : Specifies the maximum amount a given output will be true before switching to another output.
**Feedback** : A Boolean input, to provide 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 will show alarm, and the LeadLagCycles switches to the next controlled output. Setting this value to true (and not linking) disables this alarm feature.
**Out A - P** : Boolean outputs, each typically linked to a BooleanWritable control point with a DiscreteTotalizerExt. Outputs are typically used to control loads of some type, such as 2 or more pumps.
**Cycle Count A - P** : These are Integer inputs that are used for cycle count feedback for the corresponding Out A - P. These inputs will typically be linked to the ChangeOfStateCount property of the DiscreteTotalizerExt that is measuring the cycles of the corresponding Out A - P.

### LeadLagRuntime 

LeadLagRuntime provides lead-lag control of from 2 to 16 loads 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 will be active at a time based on runtime.
Setup of the object involves the following as follows:

**In** : A Boolean input that controls whether any control device should be on. If this input is true, one of the outputs will be active based on runtime.
**Number Outputs** : Specifies the number of devices (outputs) that are controlled.
**Max Runtime** : Specifies the maximum amount a given output will be true before switching to another output.
**Feedback** : A Boolean input, to provide 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 will show alarm, and the LeadLagRuntime switches to the next controlled output. Setting this value to true (and not linking) disables this alarm feature.
**Out A - P** : Boolean outputs, each typically linked to a BooleanWritable control point with a DiscreteTotalizerExt. Outputs are typically used to control loads of some type, such as 2 or more pumps.
**Runtime A - P** : These are inputs that are used for runtime feedback for the corresponding Out A - P. These inputs will typically be linked to the ElapsedActiveTime property of the DiscreteTotalizerExt that is measuring the runtime of the corresponding Out A - P.

**LoopPoint**

The LoopPoint implements a simple PID control loop, and is available in the iSMA_control palette. Loop objects provide closed-loop PID control (proportional, integral, derivative) at the controller level. Independent gain constants allow the loop to be configured as P-only, PI, or PID. Setup of the LoopPoint component involves setting the following properties:

**Loop Enable** : Setting this input to true will enable the PID loop algorithm to execute at the rate selected by the Execute Time property. Setting this input to false will force the PID loop output to a value dependent on the selection in the Preset property.

**Controlled Variable** : Input for the controlled parameter (for example, space temperature). This input must be valid for this object to function.

**Setpoint** : Input for the setpoint value (for example, space temperature setpoint). This input must be valid for this object to function.

**Execute Time** : Controls the execution frequency for the PID algorithm, where the default value is 1 second.

**Loop Action** : Determines whether the control algorithm is direct or reverse acting.

Loops setup for direct acting mode increase the loop output as the value of the controlled variable becomes greater than the setpoint value. In a temperature loop, this is typically considered to be a cooling application. Loops setup for reverse acting mode increase the loop output as the value of the controlled variable becomes less than the setpoint value. In a temperature loop, this is typically considered to be a heating application.

**Preset** :

**Max Value** sets the loop output value to the Maximum Output property value.

**Min Value** sets the loop output value to the Minimum Output property value.

**Zero** sets the loop output value to a zero (0.0) value.

**Proportional Constant** : Defines the value of the proportional gain parameter used by the loop algorithm. Used to set the overall gain for the loop. A starting point for this value is found by output range/throttling range.

**Integral Constant** : Defines the integral gain parameter, in repeats per minute, used by the loop algorithm. Also called reset rate. Acts on magnitude of the setpoint error. A typical starting point is 0.5.

**Derivative Constant** : Defines the derivative gain parameter, in seconds, used by the loop algorithm. Acts on the rate of change of the setpoint error.

**Bias** : Defines the amount of output bias added to the output to correct offset error, normally used only used with proportional control.

**Maximum Output** : Defines the maximum output value that the loop algorithm can produce.

**Minimum Output** : Defines the minimum output value that the loop algorithm can produce.

**RateOfChange**

RateOfChange Component provides the Rate of Change of Input based on the deviation & Offnormal High & Low values.

**ReheatSequence**

ReheatSequence will provide a Linear Sequence of up to 4 loads based on configurable thresholds. Sets an output true if the "in" value is greater than corresponding threshold, and returns the ouput to false if the "in" value is less than threshold minus the hysteresis value.

outA := true when in >= thresholdA
outB := true when in >= thresholdB

outC := true when in >= thresholdC
outD := true when in >= thresholdD

**SequenceBinary**

The SequenceBinary component provides sequenced weighted "staging" control of 2 to 10 loads based upon the numeric Input value (0--100). It can be used to support applications that need to sequence 2 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.
A similar object is the SequenceLinear, which uses a rotating method (vs. weighted) for sequencing. SequenceBinary is available in the HVAC folder of the iSMA_control palette.

Table 5 illustrates how, by controlling 3 loads, eight unique levels of control can be achieved:

| Control Signal (In) % | OutC (4kw load size) | OutB (2kw load size) | OutA (1kw load size) | Stage Hysteresis |
|---|---|---|---|---|
| 100 | On | On | On | 14.3 |
| 85.7 | 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 |

Setup of the SequenceBinary object involves the following properties:

**In** : Input property that is used to determine the number of stages that should currently be On.
**In Minimum** : Value of the input that produces all outputs off.
**In Maximum** : Value of the input that produces all outputs on.
**Number Outputs** : This object can be configured to support 2 to 10 outputs or stages.
**OutA - OutJ** : These are boolean values that can be used to control 2 to 10 loads. The number of outputs used is defined by the Number Outputs property.
**Desired Stages On** : Read-only property that indicates the calculated number of stages that should be on based on the In property.
**Current Stages On** : Read-only property that indicates the current number of stages that are currently on. Normally the Current Stages On and the Desired Stages On will be the same. They will be different when going through a transition.

**SequenceFailover**

SequenceFailover component is used to cascade and sequence/stage loads based on the stageStatus, cascadeMsgIn and failStatus inputs.

**SequenceLinear**

SequenceLinear provides sequenced rotating "staging" control of 2 to 10 loads based upon the numeric Input value (0--100). A similar object is the SequenceBinary, which uses a weighted method (vs. rotating) for sequencing.
The SequenceLinear component can be used to support applications that need to sequence 2 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 Numeric, is used to control the number of stages that should be on. The input range is defined by the InMinimum and InMaximum properties. SequenceLinear is available in the iSMA_control palette.
On and Off setpoints are calculated for each stage by the following Table 5 formulas (this assumes there are 5 outputs defined):

Table 5. SequenceLinear On / Off calculation formulas

| | 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 |

Setup of the SequenceLinear object involves configuring the following properties:
**In** : Input property that is used to determine the number of stages that should currently be On.
**In Maximum** : Value of the input that produces all outputs on.
**In Minimum** : Value of the input that produces all outputs off.
**Number Outputs** : This object can be configured to support 2 to 16 outputs or stages.
**OutA - OutP** : These are boolean values that can be used to control 2 to 16 loads. The number of outputs used is defined by the Number Outputs property.
**Desired Stages On** : Read-only property that indicates the calculated number of stages that should be on based on the In property.
**Current Stages On** : Read-only property that indicates the current number of stages that are currently on. Normally the Current Stages On and the Desired Stages On will be the same. They will be different when going through a transition.
**Next Stage On** : Read-only property that indicates the next stage that will be turned on if needed. This is primarily used when the Mode is selected to be Rotating.
**Next Stage Off** : Read-only property that indicates the next stage that will be turned off if needed. This is primarily used when the Mode is selected to be Rotating.
**Rotate Time** : This configuration property specifies the amount of time that the outputs will remain in a fixed configuration before the outputs are shifted to the next configuration.
**Rotate Timer Active** : Read-only property that indicates that the rotate timer is active.

**Thermostat**
Thermostat component provides the output control based on the input (process) and the set point value.

**Set Point** : Desired/target value.
**Cut In Offset** : Defines the differential value between Controlled Variable and SetPoint to determine the Thermostat output on state. A positive CutInOffset value means greater than SetPoint, and a negative CutInOffset value means lower than SetPoint during comparison. For cooling control, use positive value and negative value for heating control.
**Cut Out Offset** : Defines the differential value between Controlled Variable and SetPoint to determine the Thermostat output off state. A positive CutOutOffset value means greater than SetPoint, and a negative CutOutOffset value means lower than SetPoint during comparison. For cooling control, use negative value and positive value for heating control.

**Tstat**

Tstat provides basic thermostatic (On/Off) control with a Boolean Out property and Numeric inputs for controlled variable (Cv), setpoint (Sp), and differential (Diff).

## Latch components

Latch components allow you to capture an input value by using either the component's Clock property. "Latching" means setting the value of the latch component "Out" property to whatever the value of the latch component "In" property is at the time that the "latch" occurs. The value of the latch component "In" property is ignored at all times other than the when a latch occurs.

**BooleanLatch**

BooleanLatch provides a latch for a boolean input, and is found in the iSMA_control palette. Any latch that is invoked 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.

Latch components have the following properties that are common to all latch component data types:

**Clock** : This is a boolean property that has either a True or False state for all latch components. This property "latches" the input property to the output property on the "rising edge". This means that a single input property is captured and sent to the output property at the instant that the Clock status changes from a False to a True state and NOT when the property changes from a True to a False state.

**Out** : This standard component property provides the actual latched value that is captured from the input property at "latch" time. Link to this property to display the value on a graphic or to process the value with another component.

**In** : This is the standard component input property that you link into from a data source. For example, you can link into this property from a control point or a Schedule output.

**IntegerLatch**

IntegerLatch provides a latch for a integer input, and is found in the iSMA_control palette. Any latch that is invoked 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.

Latch components have the following properties that are common to all latch component data types:

**Clock** : This is a boolean property that has either a True or False state for all latch components. This property "latches" the input property to the output property on the "rising edge". This means that a single input property is captured and sent to the output property at the instant that the Clock status changes from a False to a True state and NOT when the property changes from a True to a False state.
**Out** : This standard component property provides the actual latched value that is captured from the input property at "latch" time. Link to this property to display the value on a graphic or to process the value with another component.

**In** : This is the standard component input property that you link into from a data source. For example, you can link into this property from a control point or a Schedule output.

**NumericLatch**

NumericLatch provides a latch for a boolean input, and is found in the iSMA_control palette. Any latch that is invoked 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.
Latch components have the following properties that are common to all latch component data types:

**Clock** : This is a boolean property that has either a True or False state for all latch components. This property "latches" the input property to the output property on the "rising edge". This means that a single input property is captured and sent to the output property at the instant that the Clock status changes from a False to a True state and NOT when the property changes from a True to a False state.
**Out** : This standard component property provides the actual latched value that is captured from the input property at "latch" time. Link to this property to display the value on a graphic or to process the value with another component.

**In** : This is the standard component input property that you link into from a data source. For example, you can link into this property from a control point or a Schedule output.

**SRLatch** 
Set/Reset Latch — single-bit edge-triggered data storage. The following logic applies on the false-to-true transition of S or R:

If S goes true and R does not change, then Out = true and remains true.
If R goes true and S does not change, then Out = false and remains false.
If both S and R go true on the same scan, then Out = false and remains false.

## Logic components

All 15 of the logic components process input values and provide a Boolean output. Logic object types vary by input types.

Seven types have Boolean inputs:

**And** 
And performs a logical AND on all inputs and writes the result to the out property. It is available in the iSMA_control palette. Table 1 shows the And object truth table when using two inputs. Table 2 shows the And object truth table if using all four inputs.

Table 1. And object truth table (2 inputs)

| In A | In B | Out |
|---|---|---|
| false | false | false |
| false | true | false |
| true | false | false |
| true | true | true |

Table 2. And object truth table (4 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 |

**LogicExpr**

LogicExpr is Binary Logic Object where various Logic Operations are be performed on one/two Boolean inputs based on the operator.

| | |
|---|---|
| out := (inA & inB) | when operator == 0 (And) |
| out := (inA \| inB) | when operator == 1 (Or) |
| out := (inA ^ inB) | when operator == 2 (Xor) |
| out := !inA | when operator == 3 (Not) |
| out := !(inA & inB) | when operator == 4 (Nand) |
| out := !(inA \| inB) | when operator == 5 (Nor) |

**Or**

Or performs a logical OR on all valid inputs and writes the boolean result to the out property. The Or is available in the iSMA_control palette. Table 3 shows the Or object truth table when using two inputs. Table 4 shows the Or object truth table when using all four inputs. NOR gate logic is accomplished by linking to a Not object.

Table 3. Or object truth table (2 inputs)

| In A | In B | Out |
|---|---|---|
| false | false | false |
| false | true | true |
| true | false | true |
| true | true | true |

Table 4. Or object truth table (4 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 |
| 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 |

**Nand** 

The Nand performs the operation out is equivalent to false if all inputs are true. It is available in the iSMA_control palette.

**Nor** 

The Nor performs the operation out is equivalent to true if all inputs are false. It is available in the iSMA_control palette.

**Not** 

The Not out simply inverts the Boolean logic value currently at the (single) object input. It is available in the iSMA_control palette.

**Xor** 

Xor performs a logical XOR on all valid inputs and writes the result to the out property. Table 6 shows the Xor object truth table when using two inputs (typical). Table 7 shows the Xor object truth table if using all four inputs.

Table 6. Xor object truth table (2 inputs)

| In A | In B | Out |
|---|---|---|
| false | false | false |
| false | true | true |
| true | false | true |
| true | true | false |

Table 7. Xor object truth table (4 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 |

Eight types have Numeric inputs:

**Comparator**
Comparator performs a Numeric Comparision of two numeric inputs and raises the respective Flags.

| | |
|---|---|
| equal | := (inA == inB) |
| notEqual | := (inA != inB) |
| greaterThan | := (inA > inB) |
| greaterThanEqual | := (inA >= inB) |
| lessThan | := (inA < inB) |
| lessThanEqual | := (inA <= inB) |

**ComparatorExpr**
ComparatorExpr is Comparator Object where various Comparator Operations are be performed on two Float inputs based on the operator.

| | |
|---|---|
| out := (inA == inB) | when operator == 0 (Equal) |
| out := (inA != inB) | when operator == 1 (NotEqual) |
| out := (inA > inB) | when operator == 2 (GreaterThan) |
| out := (inA >= inB) | when operator == 3 (GreaterThanEqual) |
| out := (inA < inB) | when operator == 4 (LessThan) |
| out := (inA <= inB) | when operator == 5 (LessThanEqual) |

**Equal** 
Equal performs the operation A == B. Numeric. NaN values are never equal.

**GreaterThan** 

GreaterThan performs the operation A > B with a boolean result. It is available in the iSMA_control palette.

**GreaterThanEqual** 
GreaterThanEqual performs the operation A >= B with a boolean result. It is available in the iSMA_control palette.

**LessThan** 
LessThan performs the operation In A < In B with a boolean result.

**LessThanEqual** 
LessThanEqual performs the operation In A <= In B with a boolean result.

**NotEqual** 
NotEqual performs the operation A != B with a boolean result. It is available in the iSMA_control palette.

## Math components

Math components process one or more Numeric input values and provide a Numeric output. Each component type provides a specific math function like Add, Average, Divide, Minimum, Maximum, Reset, AbsValue, and so on.
Math object types vary by number of inputs used, in addition to math operation.

The following Math types perform an operation on one to multiple inputs:

**Add** 
Add performs the operation out := (InA + InB + InC + InD). The Add is available in the iSMA_control palette.

**MathExpr** 
MathExpr is Object where various Mathematical & Trigonometric Operations can be performed on one/two Numeric inputs based on the operator.

| | |
|---|---|
| out := fabs (in) | when operator = 0 (AbsValue) |
| out := inA + inB | when operator = 1 (Add) |
| out := acos (in) | when operator = 2 (ArcCosine) |
| out := asin (in) | when operator = 3 (ArcSine) |
| out := atan (in) | when operator = 4 (ArcTangent) |
| out := cos (in) | when operator = 5 (Cosine) |
| out := inA / inB | when operator = 6 (Divide) |
| out := e ^ in | when operator = 7 (Exponential) |
| out := inA! | when operator = 8 (Factorial) |
| out := log10 (in) | when operator = 9 (LogBase10) |
| out := ln (in) | when operator = 10 (LogNatural) |
| out := inA % inB | when operator = 11 (Modulus) |
| out := inA * inB | when operator = 12 (Multiply) |
| out := -in | when operator = 13 (Negative) |
| out := inA ^ inB | when operator = 14 (Power) |
| out := round (in) | when operator = 15 (Round) |
| out := sin (in) | when operator = 16 (Sine) |
| out := sqrt (in) | when operator = 17 (SquareRoot) |
| out := inA - inB | when operator = 18 (Subtract) |
| out := tan (in) | when operator = 19 (Tangent) |
| out := trunc (in) | when operator = 20 (Truncate) |

**Maximum** 

Maximum determines the maximum value of valid inputs and writes that value to out. Out := max (InA,InB, InC, InD)

**Minimum** 

Minimum determines the minimum value of valid inputs and writes that value to out. Out := min (InA, InB, InC, InD).

**MinMaxAvg** 


MinMaxAvg has 5 Numeric output slots that provide the current minimum, maximum, count, sum and average values of 2 to 10 linked Numeric inputs. It is available in iSMA_control palette.

**Multiply** 

Multiply performs the calculation Out := InA * InB * InC * InD. The Multiply is available in the iSMA_control palette.

The following Math types perform an operation using two inputs:

**Divide** 

Divide performs the operation out := (in A / in B). If either input is Numeric.NaN, the output will be Numeric.NaN.

**Modulus** 

Modulus provides a modulus operation based on values at its two Numeric 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). Modulus is available in the iSMA_control palette.

**Power** 

Power performs the operation out := (InA ^ InB) or a raised to the InB power. The Power component is available in the iSMA_control palette.

**Subtract** 

Subtract performs the operation out := (InA - InB). If either input is Numeric.NaN, the output will be Numeric.NaN. It is available in iSMA_control palette.

The following Math types perform an operation on a single input:

**AbsValue** 

AbsValue performs the operation out := abs (In) (absolute value of In). The AbsValue is available in the iSMA_control palette.

**ArcCosine** 

ArcCosine performs the operation out := acos (inA). It is available in the iSMA_control palette.

**ArcSine** 

ArcSine performs the operation out := asin (inA). The ArcSine is available in the iSMA_control palette.

**ArcTangent** 

ArcTangent performs the operation out := atan (inA). The ArcTangent is available in the iSMA_control palette.

**Cosine**

Cosine performs the operation out := cos (in A). The Cosine is available in the iSMA_control palette.

**Exponential**

Exponential performs the operation out := e ^ inA (e raised to the inA power).

**Factorial**

Factorial provides a factorial math output, based upon the value present at its Numeric 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.

**LogBase 10**

LogBase10 performs the operation out := log10 (inA) (log base 10 of inA).

**LogNatural**

LogNatural performs the operation out := ln (inA) (log base e of inA).

**MinMaxAverage**

MinMaxAverage has 5 Numeric output slots that provide the current minimum, maximum, count, sum and average values of from a linked Numeric input. It is available in iSMA_control palette.

**Negative**

Negative simply converts any input numeric to a negative output value. Negative is available in the iSMA_control palette.

**Reset**

This component performs a linear "reset" on the inA value. Reset is available in the iSMA_control palette.

Reset operation is defined by the following four slots:

Input Low Limit -- must be less than the Input High Limit
Input High Limit -- must be greater than the Input Low Limit
Output Low Limit -- may (or may not) be greater than the Output High Limit
Output High Limit -- may (or may not) be greater than the Output Low Limit

For example, a Reset object is used to establish a hot water control setpoint, based on the outside air temperature at inA. 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

Whenever 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.

**Round**

Round performs the Mathematical operation of returning the nearest Integer, rounding away from zero in the halfway cases.

out := round (in)

**Sine** 

Sine performs the operation out := sin (InA). It is available in iSMA_control palette.

**SquareRoot** 

SquareRoot performs the operation out := sqrt (InA) (square root of InA). It is available in iSMA_control palette.

**Tangent** 

Tangent performs the operation out := tan(InA). It is available in iSMA_control palette.

**Truncate** 

Truncate performs the Mathematical operation of returning the nearest Integer, not greater in magnitude than the Input Float.

out := trunc (in)

## Select components

A select object allows one of multiple inputs to be selected (passed to the output) upon selection by the value at its "Select" (Integer) input. From 3 to 10 inputs can be specified.
Note that all select objects require an integer input to perform the selection--the three select object types differ only by the type of input data selected and passed to the "Out" slot.

**BooleanSelect** 

A BooleanSelect object allows one of multiple Boolean inputs to be selected (passed to the output) upon selection by the value at its "Select" (Integer) input. From 3 to 10 inputs can be specified.

Note that all select objects require an integer input to perform the selection by the type of input data selected and passed to the "Out" slot.

**IntegerSelect** 

An IntegerSelect object allows one of multiple Integer inputs to be selected (passed to the output) upon selection by the value at its "Select" (Integer) input. From 3 to 10 inputs can be specified.

Note that all select objects require an integer input to perform the selection by the type of input data selected and passed to the "Out" slot.

**NumericSelect** 

A NumericSelect object allows one of multiple Numeric inputs to be selected (passed to the output) upon selection by the value at its "Select" (Integer) input. From 3 to 10 inputs can be specified.

Note that all select objects require an integer input to perform the selection by the type of input data selected and passed to the "Out" slot.

## Switch components

A switch object allows one of two inputs to be selected (passed to the output) upon selection by the value at its "Select" (Boolean) input.

Note that all switch objects require an boolean input to perform the selection--the three switch object types differ only by the type of input data selected and passed to the "Out" slot.

**BooleanSwitch** 

A BooleanSwitch object selects between two Boolean inputs based upon the boolean value at the Boolean input 'In Switch'.

Note that all select objects require an boolean input to perform the selection by the type of input data selected and passed to the "Out" slot.

**IntegerSwitch** 

A IntegerSwitch object selects between two Integer inputs based upon the boolean value at the Boolean input 'In Switch'.

Note that all select objects require an boolean input to perform the selection by the type of input data selected and passed to the "Out" slot.

**NumericSwitch** 

A NumericSwitch object selects between two Numeric inputs based upon the boolean value at the Boolean input 'In Switch'.

Note that all select objects require an boolean input to perform the selection by the type of input data selected and passed to the "Out" slot.

## Timer components

Timer Components include 2 delay types, and a OneShot and a Timer component.

**BooleanDelay**

The BooleanDelay component provides a way to delay the change of a boolean "out" property value by configuring an associated "Delay" property. Delay properties are provided for 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 boolean input. Both delay times are configurable in terms of hours, minutes and seconds. It is available in the iSMA_control palette.

BooleanDelay component properties include the following:

**In** : Typically, you set this property by linking a boolean out value into it. You can manually configure the default state to be true or false, so that when no value is linked into this property, the default value is used. This property value is passed to the Out (after any On Delay or Off Delay) whenever there is a change in this property.

**On Delay** : This property allows you to set the amount of time (in hours, minutes, and seconds) that you want to expire before sending a true (On) value to the Out property. Time begins to expire at the moment that a change in the In property occurs (a transition from false or null to true).

**Off Delay** : This property allows you to set the amount of time (in hours, minutes, and seconds) that you want to expire before sending a false (Off) value to the Out property. The time begins at the moment that a change in the In property occurs (a transition from True to False or False to true).

**On Delay Active** : This read-only property shows whether or not the On Delay time is actively counting down to expiration. This (normally false) value changes to true anytime that a transition from false to true occurs at the In property and stays at true until any

Off Delay time is expired. If the On Delay value is set to "0", then this value does not change to true.

**Off Delay Active** : This read-only property shows whether or not the Off Delay time is actively counting down to expiration. This (normally false) value changes to true anytime that a transition from true to false occurs at the In property and stays at true until any Off Delay time is expired. If the On Delay value is set to "0", then this value does not change to true.

**Out** : This property has true, false options available. These values are set at the end of any On Delay or Off Delay to reflect the In property value.

**OneShot** 

The OneShot component provides a single, temporary, 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 Out property value is set to True and the Out Not property value is set 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.

The following types of properties are used in the OneShot component:

**In** : Typically, you set this property by linking a boolean Out value into it. You can manually configure the default state to a boolean value, so that when no value is linked into this property, the default value is used. This property value is passed to the component's Out property for the amount of time set in the Time property.

**Time** : The value of this property determines how long the Out and Out Not properties hold their "one-shot" values.

**Out** : This property value displays the current value that 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 (False) value.

**Out Not** : This property has true or false options available. The Out value change 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.

**NumericDelay**

The NumericDelay component provides a way to delay the change of a numeric "out" property value by configuring an associated "Delay" property. The delay applies to any change at the component's numeric input. The delay time is configurable in terms of hours, minutes and seconds. It is available in the iSMA_control palette.

NumericDelay component properties include the following:

**In** : Typically, you set this property by linking a numeric out value into it. You can manually configure the default state to be true or false, so that when no value is linked into this property, the default value is used. This property value is passed to the Out (after Delay) whenever there is a change in this property.

**Delay** : This property allows you to set the amount of time (in hours, minutes, and seconds) that you want to expire before sending the In value to the Out property. Time begins to expire at the moment that a change in the In property occurs.

**Delay Active** : This read-only property shows whether or not the Delay time is actively counting down to expiration. This (normally false) value changes to true anytime that a change in the In property and stays at true until any Delay time is expired. If the Delay value is set to "0", then this value does not change to true.

**Out** : This property is a numeric output. These values are set at the end of any Delay to reflect the In property value.

**Timer**

Timer outputs a pulse for the configured amount of time "in" is used to fire the timer:

if low, out is forced to false

if high, out = 1 until timer reaches "time" seconds

Alternatively, the pulse can be fired from the "Start Timer" action if in is not linked. Following are the properties for Timer:

**Out** : A timed pulse output.

**Run** : Used to fire the timer on transition from false -> true

**Time** : Desire duration of the output pulse.

**Left** : Remaining time before the output transition from true -> false

## Util components

Util Components range from a Counter object with boolean input and a numeric output (counts active transitions) to a logical AND/OR/XOR on the bit equivalent of the Numeric value against the bit equivalent of its Numeric "Mask" slot value.

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

**Counter**

The Counter component will count boolean inactive to active transitions. It supports counting up, counting down, presetting, and clearing. The Counter is available in the iSMA_control palette. The following sections provide more details:

The Counter component includes the following properties:

**Count Up** : This is a Boolean input. When this input makes inactive to active transition the value of the Out property increments by the Count Increment value.

**Count Down** : This is a Boolean input. When this input makes inactive to active transition the value of the Out property will be decremented by the Count Increment.

**Preset In** : This is a Numeric input which will be set in the Out property when the Preset action is invoked.

**Clear In** : This is a Numeric input which will be set in the Out property when the Clear action is invoked.

**Count Increment** : This is the value that the Out property will change for a single count up or count down active transition.

The Counter component includes the following actions:

**Preset** : Preset action when invoked, the value of the Out property will be set to presetIn value.

**Clear** : Clear action when invoked, the value of the Out property will be set to clearIn value.

**Multivibrator**

MultiVibrator provides an oscillating binary pulse output (Boolean) with a period configurable from 1s, and a duty cycle configurable from 0 to 100%. It is available in the iSMA_control palette.

**NumericBitAnd**

NumericBitAnd performs a logical AND on the bit equivalent of the Numeric "In" value against the bit equivalent of its Numeric "Mask" slot value. It may be useful in cases where boolean information is mapped into integer values. It is available in the iSMA_control palette.

**NumericBitOr**

NumericBitOr performs a logical OR on the bit equivalent of the Numeric "In" value against the bit equivalent of its Numeric "Mask" slot value. It may be useful in cases where boolean

information is mapped into integer values. It is available in the iSMA_control 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 parameters state (0 = false, 1 = true). The NumericBitOr object converts a Numeric 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 will result 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.

**NumericBitXor** 
NumericBitXor performs a logical XOR on the bit equivalent of the Numeric "In" value against the bit equivalent of its Numeric "Mask" slot value. It may be useful in cases where boolean information is mapped into integer values. It is available in the iSMA_control palette.
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 parameters state (0 = false, 1 = true). The NumericBitXor object converts a Numeric 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.

**Ramp** 
Ramp provides a Numeric Out with a linear ramping output. Slots define the Period, Amplitude and Offset. It is available in iSMA_control palette.

**Random** 
This component can be used to generate 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. It is available in the iSMA_control palette.
Setup of the Random component involves setting the following properties:

**Multiplier** : This is a double value that is used to multiply by the random number (the random number is >=0.0 but <1.0). The multiplier is set to 1.0 by default.

**Offset** : This is the positive or negative distance from zero that the wave's amplitude is centered on. The default offset value is 50.

**SineWave** 
SineWave generates a sine wave as a Numeric out. It is available in iSMA_control palette.

**Frequency** 
Frequency object calculates a pulse input frequency.

**Hysteresis** 
Hysteresis sets on/off trip points to an input variable. There are two internal floats called Rising Edge and Falling Edge which are configurable.

If risingEdge > fallingEdge, then out behaves "normally", ie
- out := true when in rises above risingEdge
- out := false when in falls below fallingEdge

If risingEdge < fallingEdge, then out behaves "inverted", ie
- out := false when in rises above fallingEdge
- out := true when in falls below risingEdge

If risingEdge == fallingEdge, this object behaves as a simple comparator,
- out := true when in > Rising Edge

**Limiter** 

Limiter object restricts the output based on the input between lowLimit and highLimit.
HighLimit and LowLimit are configurable floats :

out := highLimit when in > highLimit
out := lowLimit when in < lowLimit
out := in when lowLimit < in < highLimit

**Linearize** 

Linearize — piecewise linearization of a float.
For piecewise linearization of a nonlinear input, there are ten pairs of x,y parameters that must be configured into this component. The x,y pairs indicate points along the input curve. For an x value of the input, there should be a corresponding y value of the output. For input values between these points, the component will estimate the output based upon the linear equation:

Converts a table of values into a curve using linear interpolation between the values.
The x,y pairs indicate points along the input curve, for an x value of input there should be a corresponding y value of output.
Individual slope/intercept constants are computed between the x's and y's using the formula y = mx + b, where m = ym - yn/xm - xn.

If in is not in the range of x0 to x9, then output is set to "nan"

Note that slope may be positive or negative, and is indicated by comparison of x1 and x0.
Positive if x1 > x0
Negative if x1 < x0

out := (m * in) + b where m is the slope between the adjacent points and b is the Y intercept.

**TempConversion** 

TempConversion is a Converter object to convert Temperature from one unit to another.

out := in when in = celsius & out = celsius
out := (in - 32.0) * (5.0/9.0) when in = celsius & out = fahrenheit
out := in + 273.0 when in = celsius & out = kelvin

out := (in * 1.8) + 32.0 when in = fahrenheit & out = celsius
out := in when in = fahrenheit & out = fahrenheit
out := (in * 1.8) + 32.0 + 273.0 when in = fahrenheit & out = kelvin

out := in - 273.0 when in = kelvin & out = celsius
out := ((in - 273.0) - 32.0) * (5.0/9.0) when in = kelvin & out = fahrenheit
out := in when in = kelvin & out = kelvin

**UpDown** 

The UpDown component will count based on the countIncrement property. It supports counting up, counting down, presetting, and clearing.

out := out + countIncrement when mode = true (Up Mode)
out := out - countIncrement when mode = false(Down Mode)
out := No Change when mode = null (Disable)
out := presetValue when preset action is fired
out := 0.0 when clear action is fired