SmartOperator

Description

Smart Operators make it possible to synthesize several points from a query. The syntheses can be of various types depending on the SmartOperator used:

- On the status (SmartStatus)
- On numeric values (SmartAdd, SmartAverage, SmartMin, SmartMax)
- On Boolean values (SmartAnd, SmartOr)
- On operating times (SmartTimeTotalizer)
- On event counters (SmartCountTotalizer)
- On integrated values (SmartNumericTotalizer)

The advantage of SmartOperator lies in the optimization of the JACE resources because it only executes the initialization query according to the parameterization. The components considered by the Smart Operator (and which must be selected by the initialization query) are as follows:

- NumericPoint
- BooleanPoint
- EnumPoint
- StringPoint
- All other SmartOperators
- In the case of SmartStatus, the components with a status slot

The Smart Operator Ext extension in the component takes care of the calculations and then returns the result to the ControlPoint (StringPoint, Numeric Point or BooleanPoint) of where it is located. The out slot of the Smart Operator also contains the status synthesis of the points.

Implementation

- To use the Smart Operator component, drag and drop a Smart Operator from the btibToolkit palette to anywhere in the station.

(To use the extension in an existing point, drag and drop the Smart OperatorExt extension present to inside the Smart Operator).

- Enter the point selection query in the initQuery slot (⚠️ Only the legitimate SmartOperator targets will be taken into account).

Example:

- station:|slot:/Drivers|bql: select * from control:ControlPoint
- station:|slot:/Drivers|bql: select * from control:NumericPoint where floor = 'Floor 2'
- slot:../../|bql: select * from control:BooleanPoint

SmartMin (Numeric Point)

Facets	units=null,precision=1,min=-inf,max=+inf
Proxy Ext	null
Out	- {disabled,null}
SmartMinExt	- {disabled,null} (value = nan)
Advanced Config	Advanced Config
Init On Start	false
Init On Clone	false
Display Counts On Parent	false
Invalid Status	disabled fault down alarm stale overridden null unackedAlarm
Propagate Nan	true
Exclude Infinite Values	true
Selection	Control Point Only
Auto Conversion	false
For Smart Operators Of Same Type Considered As	Single Point
Enabled	false
Init Query	station:|slot:/Drivers|bql:select * from control:NumericPoin
Min Update Time	+00000h 00m 15s
inputCount	0 {ok}
inputSmartOperatorCount	0 {ok}
totalPointCount	- {null}
invalidPointCount	- {null}
invalidSmartOperatorCount	- {null}
result	- {disabled,null}
value	nan

SmartStatus
String Point

Out	- {fault,alarm,stale,overridden,null,unacked
inputCount	14 {ok}
inputSmartOperatorCount	0 {ok}
totalPointCount	14 {ok}
invalidPointCount	0 {ok}
invalidSmartOperatorCount	0 {ok}
result	- {fault,alarm,stale,overridden,null,unacke
value	singlePoint
nanCount	0 {ok}
alarmCount	1 {ok}
unackedAlarmCount	3 {ok}
downCount	0 {ok}
overriddenCount	1 {ok}
faultCount	4 {ok}
disabledCount	0 {ok}
undefinedCount	4 {ok}
staleCount	1 {ok}
okCount	1 {ok}
alarm	true {ok}
unackedAlarm	true {ok}
down	false {ok}
overridden	true {ok}
fault	true {ok}
disabled	false {ok}
undefined	true {ok}
stale	true {ok}
allOk	false {ok}

Properties

Extension properties

- *Enabled*: Allows the Smart Operator to be activated

⚠ If the Smart Operator is not initialized an error message appears. Check the error messages in the Application Director to understand the reason for the failure.

- *InitQuery*: The query allows selection of the ControlPoints on which the synthesis is to be performed. The request may be bql or neql. The selected components must be either controlPoints, SmartOperators or, in the case of SmartStatus, components with a status slot.

⚠ It is possible to define a relative query. This query is expressed relative to the extension (SmartMinExt for example). It is therefore necessary to go to the parent twice (slot:../../|bql:) to retrieve elements in the same component as the SmartOperator.

- *MinUpdateTime*: This is the minimum time between two runs of the Smart Operator. When there is a change to an entry point, the Smart Operator updates itself and starts a "timeout" period during which all changes are neutralized and the Smart Operator is only updated at the end of this period (including all changes during the "timeout").

⚠ This property makes it possible to optimize the Smart Operator's processing. A null value disables this function, but in this case any change to an input property of the Smart Operator triggers the calculation, so if the synthesis includes many points, numeric for example, it can be resource-intensive. Irrespective of the setting of this variable, it is also a good idea to create a Smart Operator tree to distribute the processing rather than having a Smart Operator connected to many points.

- *pointX*: A StatusValue that represents a ControlPoint on which the extension will work. It contains the value and the status to be taken into account by the extension.
- *smartX*: A StatusValue that represents a SmartOperator on which the extension will work. It can be considered SinglePoint or MultiplePoint depending on its "consideredAs" property.

Advanced configuration properties

- *InitOnStart*: If this property is enabled, the Smart Operator's initialization request will restart with each start of the station. ⚠ It is strongly discouraged that this option be activated when the Smart Operator is processing points from the Niagara Network.
- *InitOnClone*: If this property is enabled, duplicating a Smart Operator will automatically start its initialization request (this is particularly useful if you have a relative initQuery)
- *DisplayCountsOnParent*: Displays additional properties at the parent point. Adding this information increases the processing necessary.
- *InvalidStatus*: Allows definition of the states considered to be invalid. All states (including invalid ones) will be propagated in the state of the out slot. However, only the values whose state is valid will be taken into account. In the case of Smart Status, no values are processed, so all states are considered valid by default. This property is also used to count the number of invalid points, saved in the InvalidPointCount property (see parent point properties).
- *PropagateNan*: In the case of SmartOperators dealing with NumericPoints, propagates a NaN to the output. In other words, a NaN input forces a NaN output.
- *ExcludeInfiniteValues*: In the case of SmartOperators dealing with NumericPoints, disables the +inf and -inf values in the calculation of the output value.
- *Selection {ControlPointOnly, SmartOperatorOnly, Both}*: SmartOperators can work on other SmartOperators. This property is used specifically if you want to work only on ControlPoints that do not have a SmartOperator extension, only on points with a SmartOperator extension, or both.

⚠ In the case of Smart Status, the ControlPointOnly value also includes components with a status slot.

- *Auto Conversion*: In the case of SmartOperators dealing with NumericPoints, enables the conversion of values according to the facets of the numeric points and that of the SmartOperator. Only convertible points (those with facets) will be evaluated in the calculation. Non-convertible points will therefore be selected by the initialization request but ignored in the calculation and counted in the invalidPointCount.
- *ForSmartOperatorsOfSameTypeConsideredAs {SinglePoint, MultiplePoints}*: This property defines the behavior of this SmartOperator if it is processed by another SmartOperator of the same type. ⚠
    - As a SinglePoint: that is, the other SmartOperator sees it as a normal ControlPoint
    - As MultiplePoints: that is, the other SmartOperator sees it as a synthesis of points and will consider it as an aggregation of points

The SmartOperators with a numeric value (SmartAdd, SmartAverage, SmartMin, SmartMax) also have two additional slots:

- Min: The Minimum valid value. Any inferior value will be ignored.
- Max: The Maximum valid value. Any superior value will be ignored.

Advanced configuration actions:

- *SetAdditionalinitQueries*: Adds (or deletes) additional "initQueryX" initialization requests. They allow the selection of additional entries with different queries.

Properties reported at the parent point

General:

- *Out*: It takes the value of the Smart Operator result and the status synthesis calculated by aggregating the states of all the linked objects.
- *InputCount*: This is the number of points directly linked to the extension (the number of linked objects)
- *InputSmartOperatorCount*: This is the number of input SmartOperators (whether they are SinglePoints or MultiplePoints)
- *TotalPointCount*: This is the total number of points on which the SmartOperator works. If there are SmartOperators considered to be MultiplePoints, then they will not be counted as one point but as the number of points on which they themselves work.
- *InvalidPointCount*: This is the number of invalid points from the total number of points. A point is invalid in these four cases:
    - Its state is invalid according to the InvalidStatus property
    - Its value is NaN,

- Its value is +inf or -inf AND the ExcludeInfiniteValues property is true
- Its unit is not convertible when automatic conversion is selected

⚠ When a point is considered invalid, its value is ignored BUT its state is still added to the resulting status

If there are SmartOperators considered to be MultiplePoints, then only their invalid points will be added to the invalidPointCount.

If there are input components, they are counted as points (currently only in the case of a SmartStatus)

- *InvalidSmartOperatorCount*: This is the number of input SmartOperators (so from the SmartX entries) considered invalid.

In addition to normal validity conditions (status, value, units), SmartOperators can be considered invalid if they are disabled (property set to false).

If it is invalid, the value of the Smart Operator is not taken into account in the calculations, but its status is, unless it is disabled (set to false).

If a linked activated Smart Operator is invalid:

- If it is singlePoint, then it is considered an invalid point and enters the *InvalidPointCount* count.
- If it is multiplePoints, then it is considered an aggregation of points and the *NanCount* and *InvalidPointCount* counters are added to those of the SmartOperator.
- *Value*: This is the value of the synthesis. In the case of a SmartStatus, this is equal to the Smart Operator mode.
- *Result*: This is the result of the Smart Operator encoded in JSON format. It is a StatusString that carries the resulting state

In DisplayCountsOnParent only mode:

- *NanCount*: The number of points whose value is NaN
- *AlarmCount*: The number of points with alarm status from the total number of points
- *UnackedAlarmCount*: The number of points with unackedAlarm (unacknowledged alarm) status from the total number of points
- *DownCount*: The number of points in the down state (loss of communication) from the total number of points
- *OverriddenCount*: The number of points with an overridden status from the total number of points
- *FaultCount*: The number of points with an error status from the total number of points
- *DisabledCount*: The number of points with disabled status from the total number of points
- *UndefinedCount*: The number of points whose status is null (undefined) from the total number of points
- *StaleCount*: The number of stale points (frozen) from the total number of points
- *OkCount*: The number of points in ok status from the total number of points

⚠ It should be noted that in the case of SmartOperators working on other SmartOperators considered to be MultiplePoints, the counts from the SmartOperators "children" will be added to those of the parent.

Example: A SmartAdd has the following counters: alarmCount = 3 and faultCount = 2. Its resulting state is therefore {alarm, fault}.

A second SmartAdd (called GeneralSynthesis) works on this SmartAdd as well as on an additional point whose state is {alarm, unackedAlarm}.

If the SmartAdd is considered a SinglePoint, the GeneralSynthesis component will have the following counters: alarmCount = 2 (the SmartAdd + the point), UnackedAlarmCount = 1 (the point), faultCount = 1 (the smartAdd) and the other counters at 0.

However, if the SmartAdd is considered as MultiplePoints, the GeneralSynthesis component will have the following counters: alarmCount = 4 (the three points of the SmartAdd + the extra point), UnackedAlarmCount = 1 (the extra point), faultCount = 2 (the two points of the SmartAdd) and the other counters at 0.

- *Alarm*: True if the resulting state is an alarm
- *UnackedAlarm*: True if the resulting state is unacknowledged
- *Down*: True if the resulting state is communication loss
- *Overridden*: True if the resulting state is overridden
- *Fault*: True if the resulting state is an error
- *Disabled*: True if the resulting state is disabled
- *Undefined*: True if the resulting state is undefined (null)
- *Stale*: True if the resulting state is frozen
- *AllOk*: True if the resulting state is ok

⚠ The syntheses of the states are independent of the syntheses of the values. If a point is invalid because of its value (+inf, NaN, etc.), its state is still counted in the synthesis status.

Actions

- *Initialize*: Creates as many StatusValue (called pointX) as valid objects selected by the query and links them to the out slots of these objects. Also creates as many StatusString (called smartX) as SmartOperator selected by the query and links them to the result slots of these SmartOperator. Once the initialization is complete, the execution of the Smart Operator can be launched. (For the components with a status slot linked to a SmartStatus, it is this slot status that is linked.)

⚠ If the initialization is not successful, check the error messages in the Application Director. There is no pop-up to indicate initialization failure.

- *ForceExecution*: Forces Smart Operator execution even if the MinUpdateTime timer is active (be careful not to run the object too often if it has a lot of entries)
- *Clean*: Resets the Smart Operator by removing all related objects and resetting all of its counters to zero. The SmartOperator is then deactivated, (it will be necessary to use the initialize command if you want to recreate the linked slots)
- *Check*: In the case of numeric SmartOperators using unit conversion, this action verifies that facets of the linked points have not been modified, and if a change is detected, performs an initialization.

- SmartStatus
- SmartAdd
- SmartAverage
- SmartMin
- SmartMax
- SmartAnd
- SmartOr
- SmartTimeTotalizer
- SmartCountTotalizer
- SmartNumericTotalizer

Training

You can follow this e-learning course to practice