

Information and/or specifications published here are current as of the date of publication of this document. Tridium, Inc. reserves the right to change or modify specifications without prior notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia. Products or features contained herein are covered by one or more U.S. or foreign patents. This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc. Complete Confidentiality, Trademark, Copyright and Patent notifications can be found at: <http://www.tridium.com/galleries/SignUp/Confidentiality.pdf>. © 2011 Tridium, Inc.

JACE, Niagara Framework, Niagara AX Framework and the Sedona Framework are trademarks of Tridium, Inc.

BQL Expression component

BQL Expression component (Expr), found in the Util folder of the kitControl palette, is a multi-purpose wire sheet object for mathematical and logical operations that augments the existing math and logic components already present in the kitControl palette.

Expr can reduce the number of component instances in your station by allowing you to quickly and easily create simple logic and math statements. You do not need Java programming knowledge, a Program component, or even multiple standard components to do this. Expr does not require compilation.

Note: This component is available in a station running on a AX-3.6 and later host.

These sections describe the component and how to use it:

- “Component features” on page 1
- “What the component is not” on page 2
- “Syntax” on page 2
 - “Supported operators” on page 2
 - “Commas” on page 3
 - “Long statements” on page 3
- “Create a BQL Expression component” on page 3
 - “Mathematical expressions” on page 4
 - “Logical expressions” on page 5
 - “Component instances” on page 5
 - “More examples” on page 5
- “Handling null” on page 8
- “Troubleshooting” on page 9
- “Frequently-asked questions” on page 9
- “Document change log” on page 9

Component features

BQL Expression component supports:

- Math and logic operators
- Multiple expressions (delimited by commas) within a single component
- Dynamically-created Expr inputs and output(s) based on the expression(s)
- Automatic link conversion between different data types, for example, Double to StatusNumeric

When configuring the component, you specify all of its input slots to “Execute On Change”, such that the Expr executes upon any input change. However, the component also allows you to override this behavior by specifying a time delay between executions. This can be useful to minimize the effects of rapidly changing inputs. Other properties provide status and fault cause information.

In addition to the blank Expr component found in the Util folder of the kitControl palette, two example Expr components: ExprLogic (in the Logic folder), and ExprMath (in the Math folder) each use a single BQL expression to demonstrate how the component works.

What the component is not

BQL Expression component is not:

- A replacement for the Program component. BQL Expression component does not support `//remarks` and `//comments`.
- A replacement for other BQL statement containers that manipulate data.
- Capable of manipulating data through stored variables.
- Capable of line-by-line programming.
- Capable of handling time functions.

If your requirements exceed what can be achieved using a BQL Expression component, there may come a point when you have to consider using a Program component.

Syntax

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)

Supported operators

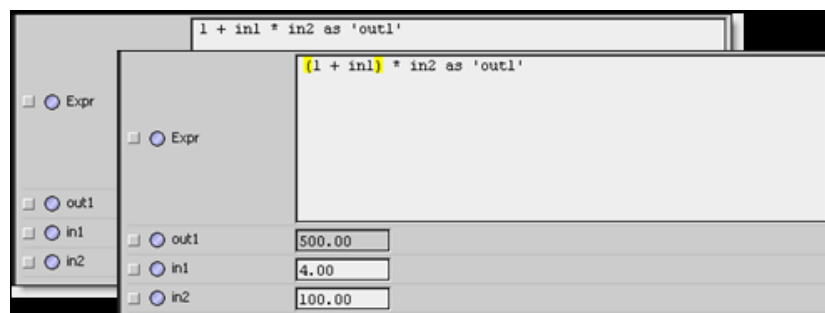
As Expr uses BQL, all the standard BQL expression syntax is available. For more information on BQL expressions, see the *NiagaraAX Developer Guide*.

Operators are processed by their *precedence*, that is “order of operation,” from first (1) to last (6).

1. `!`, `not`, `-`
logical not, numeric negation
2. `*`, `/`
multiplication, division
3. `+`, `-`
addition, subtraction
4. `=`, `!=`, `>`, `>=`, `<`, `<=`, `like`
comparisons
5. `and`, `or`
logical operators
6. `as`
result operator

You may use parentheses to override the normal precedence as illustrated in the following examples.

Figure 1 A change in precedence

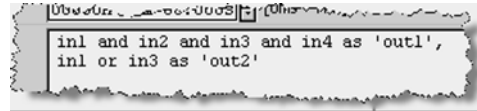


In the first expression, multiplication precedes addition. Adding the parentheses changes the precedence so that addition precedes multiplication.

Commas

If creating a Expr component with multiple expressions (output slots), use a comma to delimit expression statements from one another.

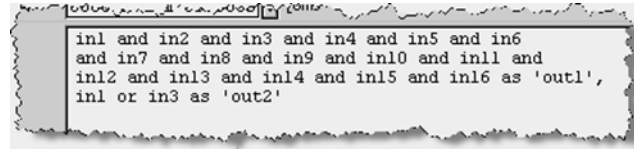
Figure 2 Example of commas



Long statements

When entering an expression, keep typing or enter a CR/LF to wrap a long statement.

Figure 3 Example of a CR/LF



The example above consists of two expression statements: the first requires three lines; the second requires only a few characters.

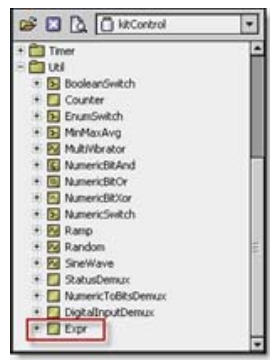
Note: Outputs require surrounding apostrophes; inputs do not.

Create a BQL Expression component

To create a BQL Expression component

- Step 1 Design your expression. This step answers the question, "What does Expr need to do?"
For example, say you wish to create a logic component with four Boolean inputs and two Boolean outputs. One output is an AND function on all inputs, the other output is an OR function on just two of the inputs.
- Step 2 Drag an Expr from the kitControl palette onto your wire sheet and give it a name.

Figure 4 kitControl components with BQL Expression (Expr) highlighted



- Step 3 Using the slot sheet view of the component, add four Boolean inputs and two Boolean outputs to the Expr and give them appropriate and unique names.

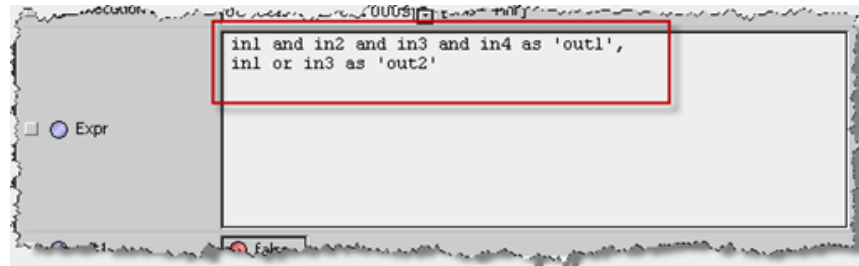
Note: When adding input slots, set (check) the **Execute On Change** flag for each one.

Figure 5 New slots (out1, out2, in1, in2, in3, in4)


Property	12	Link3	Link3	Dynamic		baja:ConversionLink
Property	13	out1	out1	Dynamic	rs	baja:Boolean
Property	14	out2	out2	Dynamic	rs	baja:Boolean
Property	15	in1	in1	Dynamic	sxL	baja:Boolean
Property	16	in2	in2	Dynamic	sxL	baja:Boolean
Property	17	in3	in3	Dynamic	sxL	baja:Boolean
Property	18	in4	in4	Dynamic	sxL	baja:Boolean


- Step 4 Using the property sheet view, configure any execution delay.
The expression will run after this delay, which is useful to throttle rapidly changing properties used as inputs.
- Step 5 Continuing on the property sheet, enter the BQL statements in the Expr field.

Figure 6 Example expression statements



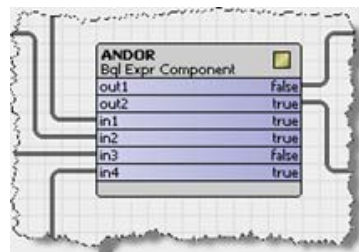
All other properties are assumed to be dynamic properties and can be added or removed on the Expr slot sheet.

- Step 6 Click **Save**, then the  “Up Level” menu bar icon to return to the parent wire sheet. The new Expr component is visible, but without any “pinned” slots for inputs and outputs.

Although optional, you can pin slots *before* linking, by right-clicking the component and selecting  **Pin Slots**. This can speed up linking, as otherwise the popup **Link** dialog appears if linking to unpinned slots. You can also use the right-click **Reorder** option to change the top-to-bottom positioning of slots.

- Step 7 Using the wire sheet view, make links to the slots and test your logic.

Figure 7 Example of an expected result



The figure above shows all slots of this example Expr component linked.

Mathematical expressions

At the heart of Expr is a BQL expression that processes inputs and outputs. You have, for example, a component with three properties: inA, inB and outAdd. To add the two inputs together you would use this expression:

```
inA + inB as 'outAdd'
```

In mathematical terms, this means:

```
inA + inB = outAdd
```

In the example, inA and inB must use the executeOnChange slot flags. When inA or inB change, Expr executes and updates outAdd.

Expr is not restricted to a limited number of properties. You can add and remove properties via Expr's slot sheet. For example, using the above, we could add another two properties called outMult and inC. The expression would now read:

```
inA + inB as 'outAdd',
inA * inB * inC as 'outMult'
```

The comma at the end of the first expression indicates that another expression follows. Use the comma to create multiple expressions that update multiple outputs.

Logical expressions

In the following example, two Boolean properties use an AND gate to output a Boolean value.

inBoolA and inBoolB is 'out'

Again, the inBoolA and inBoolB must use the executeOnChange slot flag.

Note: From Niagara 3.6 onwards, linking slots with different data types automatically inserts a conversion link between the two. For example, you may link a *StatusNumeric* property to a *Double* property. For more information, see the *Engineering Notes* document *NiagaraAX Conversion Links*.

Component instances

The kitControl palette contains multiple instances of the BQL Expression component with different default configurations.

- **Util folder**
A blank Expr is its default state.
- **Logic folder**
Provides Expr with four Boolean inputs and one Boolean output. By default, all inputs are joined using AND.
- **Math folder**
Provides Expr with four Double inputs and one Double output. By default, all inputs are mathematically added together.

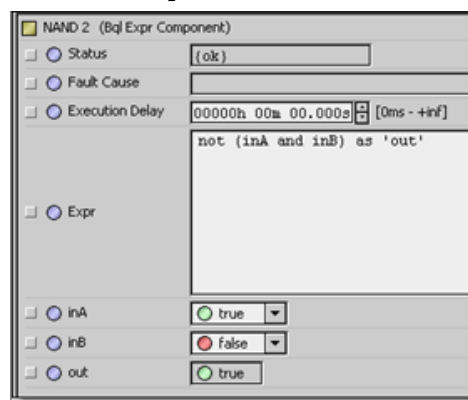
Note: Any of these components can be modified by adding and removing properties using the slot sheet.

More examples

- **Two inputs, logical AND**



- **Not two inputs**



- **Two-input addition**

The screenshot shows the configuration window for the 'ADD (Bql Expr Component)'. It includes fields for Status (set to 'ok'), Fault Cause, and Execution Delay (set to '00000h 00m 00.000s' with a range of '[0ms - +inf]'). The 'Expr' field contains the expression 'in1 + in2 as 'out1''. Below this, the output 'out1' is shown with a value of '123.00', and inputs 'in1' and 'in2' are shown with values '23.00' and '100.00' respectively.

- **Divider (two Double) properties**

The screenshot shows the configuration window for the 'DIVIDE (Bql Expr Component)'. It includes fields for Status (set to 'ok'), Fault Cause, and Execution Delay (set to '00000h 00m 00.000s' with a range of '[0ms - +inf]'). The 'Expr' field contains the expression 'in1 / in2 as 'out1''. Below this, the output 'out1' is shown with a value of '6.67', and inputs 'in1' and 'in2' are shown with values '100.00' and '15.00' respectively.

- **Subtraction**

The screenshot shows the configuration window for the 'SUBTRACT (Bql Expr Component)'. It includes fields for Status (set to 'ok'), Fault Cause, and Execution Delay (set to '00000h 00m 00.000s' with a range of '[0ms - +inf]'). The 'Expr' field contains the expression 'in1 - in2 as 'out1''. Below this, the output 'out1' is shown with a value of '-100.00', and inputs 'in1' and 'in2' are shown with values '0.00' and '100.00' respectively.

- **Expression mixture**

MIX 1 (Bql Expr Component)

Status: (ok)

Fault Cause:

Execution Delay: 00000h 00m 00.000s [0ms - +inf]

Expr: inA and inB as 'out1', inX + inY as 'out2'

inX: 10.00

inY: 20.00

out2: 30.00

inA: true

inB: false

out1: false

- **Greater and less than expressions**

GREATER OR LESS THAN (Bql Expr Component)

Status: (ok)

Fault Cause:

Execution Delay: 00000h 00m 00.000s [0ms - +inf]

Expr: in1 < 10 as 'out1', in1 > 10 as 'out2'

out1: 0.00

in1: 10.52

in2: 0.00

out2: true

GREATER OR LESS THAN (Bql Expr Component)

Status: (ok)

Fault Cause:

Execution Delay: 00000h 00m 00.000s [0ms - +inf]

Expr: in1 < 10 as 'out1', in1 > 10 as 'out2'

out1: 1.00

in1: 9.16

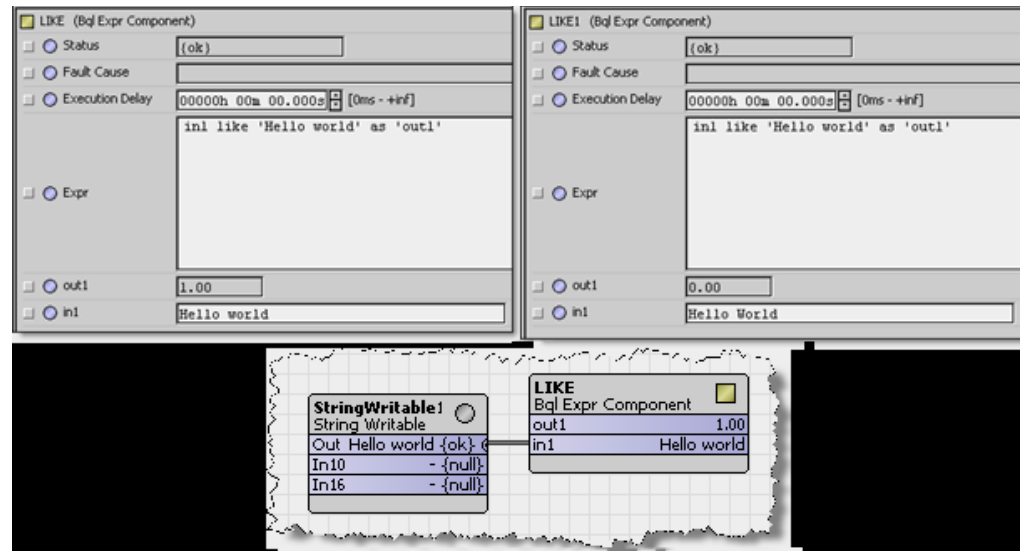
in2: 0.00

out2: false

Property	Link	Link	Dynamic	Conversion	Link
Property 9	in1	in1	Dynamic	sxL	baja:Double
Property 10	in2	in2	Dynamic	sx	baja:Double
Property 11	out1	out1	Dynamic	rs	baja:Double
Property 12	out2	out2	Dynamic	rs	baja:Boolean

The examples above each result in two outputs.
The last column on the right indicates that the two outputs are different slot types (Double and Boolean). This is legal.

- Using the “like” expression

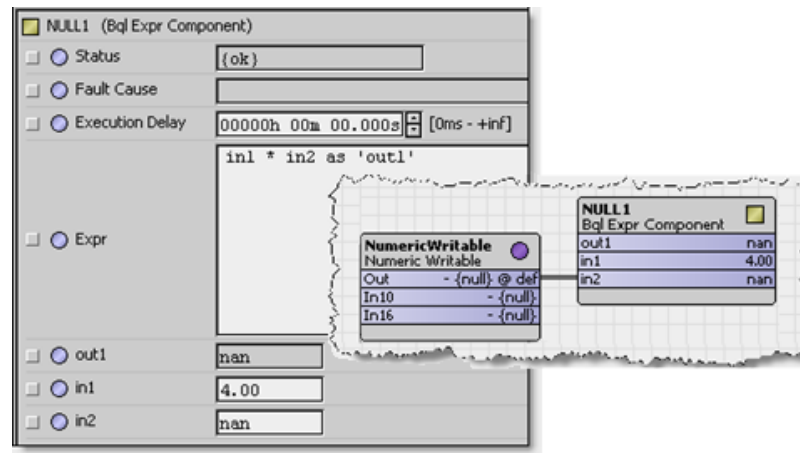


- Multiply four Double properties**
`inA * inB * inC * inD as 'out'`
- Multiply two BStatusNumeric properties to a Double output property.**
`inA.value * inB.value as 'out'`
- Negate a Double input property (if inA is 5, out becomes -5)**
`-inA as 'out'`

Handling null

This section describes expression handling of a “null” input from a linked status type output.

Figure 8 The result of a math expression with any null (or nan) input can be nan



In a *math* expression, if a Double or Float input slot is linked to a source StatusNumeric output with a “null” value, the input is evaluated as “nan” (not a number). If the expression has a Double or Float slot as output, the *result* is also nan, as shown in Figure 8. This also occurs if such an input has a nan.

Note if the input slot is an Integer or Long type, the expression ignores the null value—the last valid value is used (or if nan input, is processed as value 0). The output is some number. If an input slot is a Status-Numeric (requires expression syntax `inputSlotName.value`), a null input is seen but not processed.

In a *logic* expression, if a Boolean input slot is linked to a source BooleanNumeric output with a “null” value, the null is ignored by the expression—the last valid value is used. If the input slot is a StatusBoolean type (again, syntax `inputSlotName.value` is required), the null is seen but not processed.

The Expr component utilizes the automatic “conversion links” feature introduced in AX-3.6, such that “link from” behavior between dissimilar data types is followed. For more information, see the Engineering Notes document *NiagaraAX Conversion Links*.

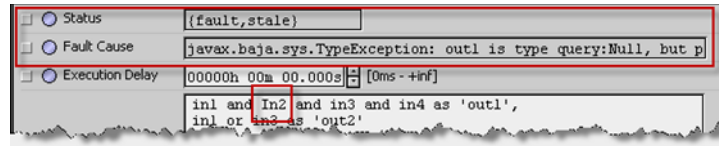
In general, if creating an Expr component for use in control logic that may have one or more “null” inputs, it is recommended that you test it to verify the desired behavior.

Note: Starting in AX-3.6, changes were also made to the various “status value to simple value” kitControl Conversion components, to allow the option of specifying an output value when the status input is null. If needed, you may wish to use one or more of these components “upstream” of the Expr component. For more details, refer to the NiagaraAX kitControl Guide section “Status value to simple value”.

Troubleshooting

If you enter the wrong syntax, or Expr recognizes that you are trying something illegal, it displays status and fault cause information.

Figure 9 Example of a fault



The information fields have the following meanings:

- **Status**
Reports problems with the expression. The expression may be invalid or one of the properties it references may not exist.
- **Fault Cause**
Gives a verbal description of expression errors.
After correcting the fault, click **Save** to restart the Expr.

Note: No compilation is required.

Frequently-asked questions

Q: How many inputs and outputs does Expr allow?

A: Expr supports as many inputs and outputs as you need. Two specific instances of Expr, located in the Logic and Math folders, each support four inputs and a single output.

Q: Does an Expr need to be compiled before it can be used on a wire sheet?

A: No, if you make a change, click Save. No compilation is required.

Q: Can I create more than one expression per component?

A: Yes. You may create as many expressions as you need, delimited by commas.

Q: Can I put comments into an Expr?

A: No. Comments are not allowed in a BQL Expression component.

Q: Can I use stored variables in an Expr?

A: No. To store variables, use a Program component.

Q: Can Expr handle time functions?

A: No. Time functions require a Program component.

Q: What types of operations does the Expr support?

A: The Expr supports math and logic operators.

Document change log

Updates (changes/additions) to this BQL Expression component document are listed below.

- Publication: April 4, 2011
Initial document.

