Binding Types in Vision

Binding is perhaps the most important concept to understand when designing a project using the Vision module. It is primarily through property bindings that you bring windows to life and have them display useful things. A binding simply links one component's property to something else.

When you initially place a component on a screen, it doesn't really do anything. Changing its properties in the Designer will make it look or act different, but it has no connection to the real world and this is what bindings adds.

Binding, as its name suggests, lets you bind a property to something else, such as

- A Tag
- The results of a SQL query executed against a remote database
- Another component's property
- An expression involving any of these things

For example, bind the Value property of an LED Display to an OPC Tag, and voilà - the value property will always be the value of that Tag - creating a dynamic display. Bindings can also work the other way, using a bidirectional binding. Bind the value of a numeric text box to a Tag, and that Tag will be written to when someone edits the value in the text box.

The power of bindings comes from the variety of different binding types that exist, and the fact that you can bind nearly any property of a component to anything else. Want its foreground to turn red when an alarm is above a certain severity? Bind its **LED Lit** property color to a Tag's **Alarms. HighestActivePriority** property. Want it to only appear if a supervisor is on shift? Bind its visible property to the result of a SQL query that joins a personnel table with a shift table. The possibilities are nearly endless.

On this page

...

Property Binding Types

A property can have one of many different types of bindings. Instead of setting a label statically, the text might change based on a PLC value or on-screen selection. There are many ways to bind your components to show values from PLCs, databases, other components, or user input. You can even bind some or all of the properties on each component. You can bind component values using:

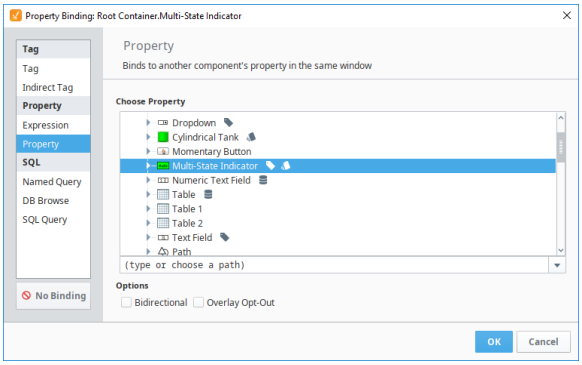
- **Property** simply binds one property to another. When that property changes, the new value is pushed into the property that the binding is setup on.
- **Tag** binds a property directly to a Tag property (typically the value) which sets up a Tag subscription for that Tag, and every time the chosen Tag property changes, the binding is evaluated, and pushes the new value into the bound property.
- **Indirect Tag** is similar to the standard Tag binding except that you can introduce any number of indirect parameters to build a tag path dynamically in the runtime.
- **Tag History** is used for Dataset type properties. It runs a query against the Tag Historian.
- **Expression** uses the simple expression language to calculate a value which can involve lots of dynamic data.
- **Named Query** executes a Named Query that had been previously created.
- **SQL Query** is a polling binding type that runs a SQL Query against any of the database connections configured in the your Gateway.
- **Database Browse** is equivalent to the SQL Query binding except that it helps write the queries for you.
- **Cell Update** enables you to easily make one or more cells inside a dataset dynamic. This is useful for components that store configuration information inside datasets like the Easy Chart.
- **Function** is a generic binding type that lets you bind a dataset property to the results of a function. It allows any of the function's parameters to be calculated dynamically via Tag and property bindings.
- **Style Customizer** is not one of the standard bindings, but changes properties to create cohesive styles based on different states.

Property Binding

Watch the Video

Property Binding - Bidirectional

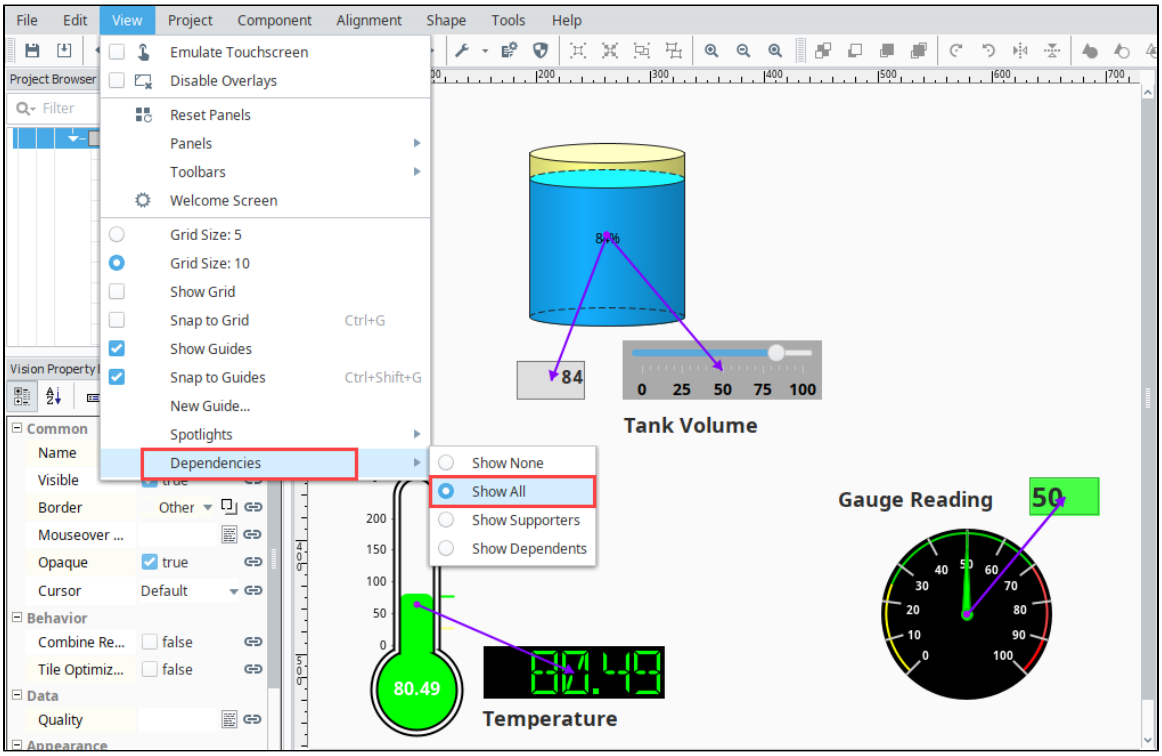
Watch the Video



Setting Up Bindings

Every component that you put on the screen has various properties that change the component's appearance and behavior. To make components do something useful, like display dynamic information or control a device register, you configure bindings on the component. It's the bindings that brings your components to life and have them do useful things. Components can be configured to do just about anything using bindings. To set up a binding on a property, simply click the Binding icon to the right of the property in the Property Editor.

In this image, bindings were set to make these random components do something. You can quickly view dependencies to determine what is linked to what by going to **View > Dependencies > Show All**. As shown below, a line is drawn from the Tank to the Slider letting you know the Tank is bound to the Slider.



Event-Based Bindings vs. Polling Bindings

While there are quite a few different binding types, they fall into two broad categories: event-based and polling. Some complex bindings can span both categories.

Event-based bindings are evaluated when the object they are bound to changes. For example, when you bind a property to a Tag, that binding listens to the Tag, and every time the Tag changes, it assigns the Tag's new value into the property that it is on. If you bind the value of a Cylindrical Tank to the value of a Slider, every time the slider changes, it fires a propertyChangeEvent. The binding is listening for this event, and when it is fired, the binding updates the tank's value. The following bindings are event-based:

- Tag and Indirect Tag bindings
- Property bindings
- Some Expression bindings
- Cell Update bindings

Polling bindings are evaluated when a window first opens, on a timer, or when they change. For example, if you bind the data property of a Table to the results of a SQL query, that query will run on a timer, updating the Table every time it executes. The following bindings are based on polling:

- Bindings that query a database, including Named Query bindings, DB Browse bindings, SQL Query bindings, and Tag History bindings
- Some Expression bindings, like runScript() or now()
- Function bindings

Many bindings can combine elements of a polling binding and event-based binding. An expression binding may combine lots of other bindings to calculate a final result. A query binding will often itself be dynamic, altering the query based on other bindings.

For example, you might have a dropdown on a window that lets the operator choose a type of product that is produced. Then you can use a query binding like the following to calculate the defect rate for the given product:

SQL - Using a Component Property Reference

```
SELECT
   SUM(defective) / COUNT(*) AS DefectRate
FROM
   production_table
WHERE
   productCode = '{Root Container.ProductPicker.selectedValue}'
```

The blue code is a property binding inside of the query binding. Every time this (event-based) binding fires, the query will run again, but will also run on a set timer based on its polling schedule. Using bindings like this, you can create highly dynamic and interactive screens with no scripting whatsoever.

Polling Options

The following are the options you can choose from for bindings that poll:

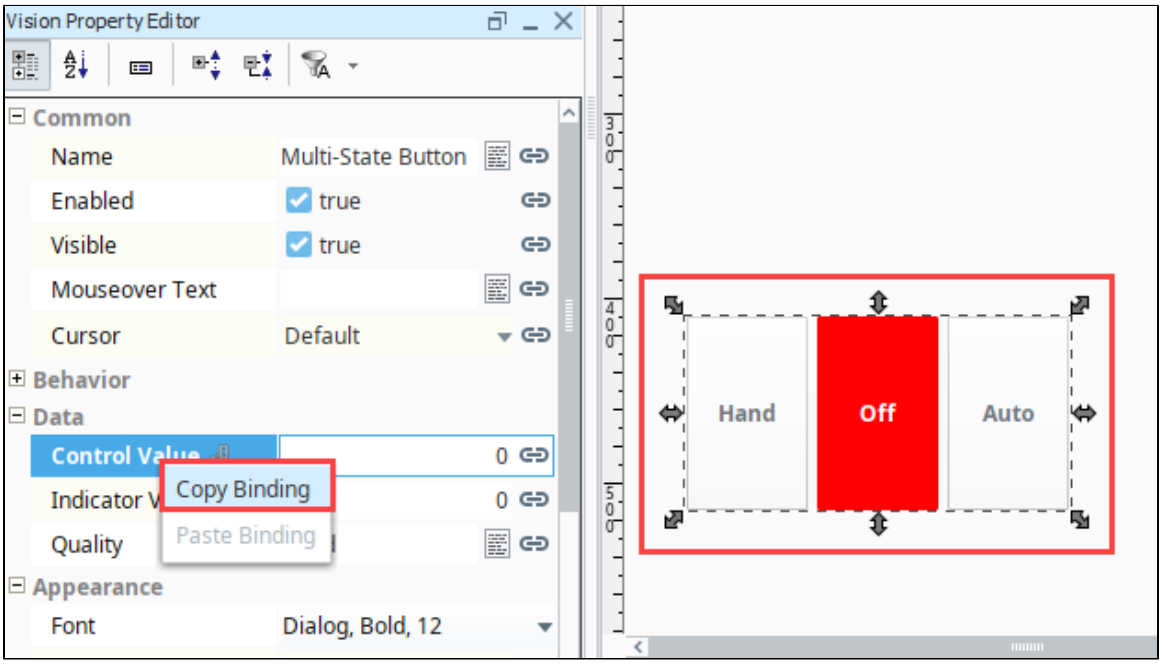
- **Polling Off**
  The query will run once when the window is opened, and again whenever a reference inside of the the binding changes.
- **Relative Rate**
  The binding will poll at the project's Base Polling Rate, which is 5 seconds by default, plus or minus the given Polling Rate.
- **Absolute Rate**
  Using this option, you can specify an absolute rate for the binding to execute at, instead of one that is based off the relative rate.

Regardless of which option is selected, polling bindings always fire when the window the component is on opens. This allows the component an opportunity to fetch an initial value.

Additionally, all three types will **always** update if the binding contains a reference to something else, such as a Tag or property value (noted with the brace-notation "{}"), and the value of that reference changes. Typically this is seen in SQL query bindings: polling can be turned off, and the query can reference a component value in a WHERE clause. When the referenced property value changes, the query will execute and retrieve new results.

Copying Bindings

When you copy a component, all bindings, scripts, etc. are copied along with it, but you can also copy a property binding from one property to another. Bindings can be copied from one property to another by right clicking on a property with a binding on it and selecting **Copy Binding**. Then, on another property, right click and select **Paste Binding** to paste the binding onto the property. This can be on the same component or a completely different component. The only prerequisite is that both property bindings must use a compatible property type. (For example, a binding that resolves to a string will not work on an integer property.)



In This Section ...