# Why does the value of my XAML Slider change when I updated its range?

devblogs.microsoft.com/oldnewthing/20200423-00

April 23, 2020

Raymond Chen

A customer found that their XAML Slider control changed its value when they updated the range.

Let's say that the Slider's initial conditions are

| | |
|---|---|
| **Minimum** | 0 |
| **Maximum** | 200 |
| **Value** | 200 |

The code then reduces the maximum to 100. The consequences of this reduction is that the value is also reduced to 100, so that the value remains clamped between the minimum and maximum.

Later, the code returns the maximum back to 200.

The customer observes that the value jumps back up to 200.

The customer was troubled by this because the value has a two-way binding, and the jump back up to 200 was causing problems. They expected the value to change only if the user grabs the slider and changes it. When the program raises the limit to 200, the value changes even though the user did nothing.

What's going on is that the Slider control remembers that the value "wants to be" 200, but the range restriction forces the value to be reduced to 100. When the restriction is lifted, the value returns to its "natural" value of 200.

This behavior is a consequence of the principle that properties can be set in any order, even if that results in temporary inconsistencies.

Allowing properties to be set in any order permits related properties to be set by binding. The order in which bound properties are set is difficult if not impossible to control, so it is a general principle that properties may be set in arbitrary order, even if the results are temporarily inconsistent. Enforcement of inconsistent properties is deferred until it is clear that the property-setting phase is complete, typically when you call a method on the object.

For example, suppose that the maximum and value start at 200.

| | |
|---|---|
| `slider.Maximum = 100;`<br>`slider.Value = 100;` | `slider.Value = 100;`<br>`slider.Maximum = 100;` |

Both of these are valid sequences of operations for setting the maximum and value to 100. If the Slider control had raised an error whenever inconsistent values were assigned, then the first version would have failed at the reduction of the maximum to 100.

Another idea would be that reducing the maximum to 100 would also reduce the value down to 100. But that has its own problems: What if the maximum is (temporarily) less than the minimum? What would you set the value to?

It is not uncommon to set the range and current value to defaults, and then adjusting depending on what case you are in. If coersion occurred immediately, then you may find yourself updating a value that will ultimately not require updating.

Specifically, consider this:

```
slider.Maximum = 100;
if (large_range) slider.Maximum = 200;
```

In the case where large ranges are enabled, the expectation is that a value of 150 is allowed. But if the value were updated immediately when the maximum was reduced to 100, then the value would be lowered to 100 and stuck there.

And that's the case the customer found themselves in. They lowered the maximum to 100, and the current value of 200 was clamped (but not updated) to the current range. When the maximum was raised back to 200, the original value of 200 was allowed to shine through.

The control just figured you were being really slow to decide what your maximum was.

So what do you do if you want the value to be 100 and stay that way, even if the maximum is later raised to 200?

Easy: Set it to 100.

Raymond Chen

**Follow**