Distinguishing between normative and positive statements to help people answer your question

devblogs.microsoft.com/oldnewthing/20141119-00

November 19, 2014

Raymond Chen

Often, we get questions from a customer that use the word *should* in an ambiguous way:

> Our program creates a widget whose flux capacitor should have reverse polarity. Attached is a sample program that shows how we create the widget with `CreateWidget`. However, the resulting widget still has a flux capacitor with standard polarity. Can you help us?

The phrase *should have reverse polarity* is ambiguous. The question could be

> We would like to create a widget whose flux capacitor has reverse polarity. Attached is a sample program that shows how to create a widget whose flux capacitor has standard polarity. How should we modify it in order to get reverse polarity?

Or the question might be

> We would like to create a widget whose flux capacitor has reverse polarity. Attached is a sample program that attempts to do so, but the resulting widget has a flux capacitor with standard polarity. The polarity flag appears to be ignored. Are are we doing something wrong, or is this a bug in Windows?

The first is a normative statement: "This is what we would like to happen." The second is a positive statement: "This is what is happening." The distinction is important because the two types of statements require very different types of responses. If have a program that does X, and you want to change it to do Y, then you're asking for help working through the Y feature, clarifying the documentation, informing you which flags you need to pass, and so on. But if you have a program that tries to do Y and fails, then you're asking for help debugging your code and possibly identifying a bug in the operating system. Being clear with your request means that you can avoid wasting a lot of time when the wrong set of people are called in to help you out. Here's another example of vague use of the word *should*:

> We're trying to do XYZ. We've been told that it is blocked for security reasons, but there should be a way to do this.

In this case, it is not clear what the customer means by the phrase *should be a way to do this*. It could be

> We're trying to do XYZ. We've been told that it is blocked for security reasons, but we think that Windows should be changed to allow our scenario. How can we file a change request with the Windows security team to make an exception for us?

Or the customer might be trying to say

> We're trying to do XYZ. We've been told that it is blocked for security reasons, but we think that there is a way to get the effect of XYZ without triggering the security issue. Can you help us find it?

Note that in both cases, the customer either failed to asked a question or made some statements and asked for nonspecific advice, which is effectly the same as not asking a question. If they had remembered to ask a question, then that question would have clarified what they intended by the word *should*. **Bonus chatter**: A physicist classmate of mine got a chuckle out of the phrase *flux capacitor* because it combines two physics terms in an impressive-sounding but mostly nonsensical way. A *capacitor* is a device which stores electric potential. In the hydraulic analogy of electricity, a capacitor is a rubber diaphragm that separates two parts of a pipe, but which "stores" water flow by stretching and "discharges" the water flow by returning to its rest position. *Flux* is cross-sectional flow per unit time. Water flux is volumetric flow rate (liters per second per square meter): it measures how vigorously the water flows across a boundary. Magnetic flux measures the strength of a magnetic field. The combination is nonsensical because the units don't match. A capacitor stores potential, whereas flux is measured in current or magnetic field strength. But if you generalize the term *capacitor* to mean "a thing that stores stuff", then a *flux capacitor* is a device which stores a magnetic field.

Such devices already exist today. They are called *magnets*.

Raymond Chen

Follow