Troubleshooting tips are not formal product documentation

devblogs.microsoft.com/oldnewthing/20060424-21

April 24, 2006

Raymond Chen

The Microsoft Knowledge Base is filled with product support tips, but be careful to understand the scope of those tips. Generally speaking, information provided in the Knowledge Base exists for troubleshooting purposes, not for program design. That's why each article lists specifically which operating system it applies to: There is no guarantee that a particular tip will work on future operating systems. This is particularly true if the tip recommends digging into a program's internal data structures, persistence formats, or files. The Knowledge Base article is for helping you get yourself out of a mess; it is not formal product documentation. Think of it as a Microsoft version of Experts Exchange. (I always wondered how many people misread that site as "Expert Sex Change".) Knowledge Base articles typically come from the product support groups. Someone might solve a problem with a customer's machine and say, "You know, I bet other people will run into this problem, too. Let me document how I fixed it to save time for the next person." They'll write up a Knowledge Base article and add it to the database. Sometimes (rarely, in my experience), they will run the article past the product group responsible for the topic area for a technical review before publishing it. In other words, articles in the Knowledge Base come with a lower "level of service" than formal product documentation; they were not necessarily approved by the product group. If it helps you solve your problem, then great! But there's no promise that it will, and neither is there a promise that the solution will work in the next version of the operating system. It's just a bunch of IT folks sharing war stories. The explanation above is also wrong. Due to the quick publishing times of the Knowledge Base compared to the formal product documentation, a product group will often take a shortcut and publish their documentation via the Knowledge Base in order to get the information out quicker. As a result, some Knowledge Base articles really are formal product documentation disguised as troubleshooting tips. Sometimes a Knowledge Base article does go through product group review, such as the pair of articles describing how to use rundll32 to debug a control panel and how to start a control panel programmatically. The programmatic method is to use the `Control.exe` program, which has been the supported mechanism since Windows 3.0. The `rundll32` technique is only for debugging purposes. If you use the `rundll32` technique in your production software, you will run into problems because you're bypassing the official entry point and going for the internal entry point. The official entry point has the compatibility hacks, knowing about the names of control panels that have been retired and redirecting them to their replacements, knowing where shell special folders have moved to, and knowing how to compensate for 64-bit programs trying to run 32-bit control panels and vice versa. But if you go straight to `rundll32`, you miss out on all this. That's why the article says that the `rundll32` method is for debugging purposes, not for production use. The bad news for you, poor reader, is that it's not clear from the text of the article which type of article you have. Is it (1) an as-is tip from a product support technician in the hopes you might find it useful, (2) an as-is tip that the product group approved as solution that they will support in the future, or (3) product documentation disguised as a support tip? Personally, I can tell the difference between {1,2} and {3} just by the tone of the article and the general sense of "Gosh, I wonder if the product team really expects to be supporting this mechanism in the future." Maybe I was just born with this skill (or perhaps developed it early on in my career as a software developer), because it seems most people can't tell the difference. Here's the question to ask: Does the proposed solution rely on what appears to be a side-effect, internal behavior, or internal data structure that a programmer would want to reserve the right to change in the future? Does it feel like a cobbled-together solution? Does it involve duct tape and string?

I'm sorry it's such a mess, but with hundreds of thousands of Knowledge Base articles in existence, it would be impossible for me to go through and tag each one. You'll have to use your best judgement.

Raymond Chen

Follow