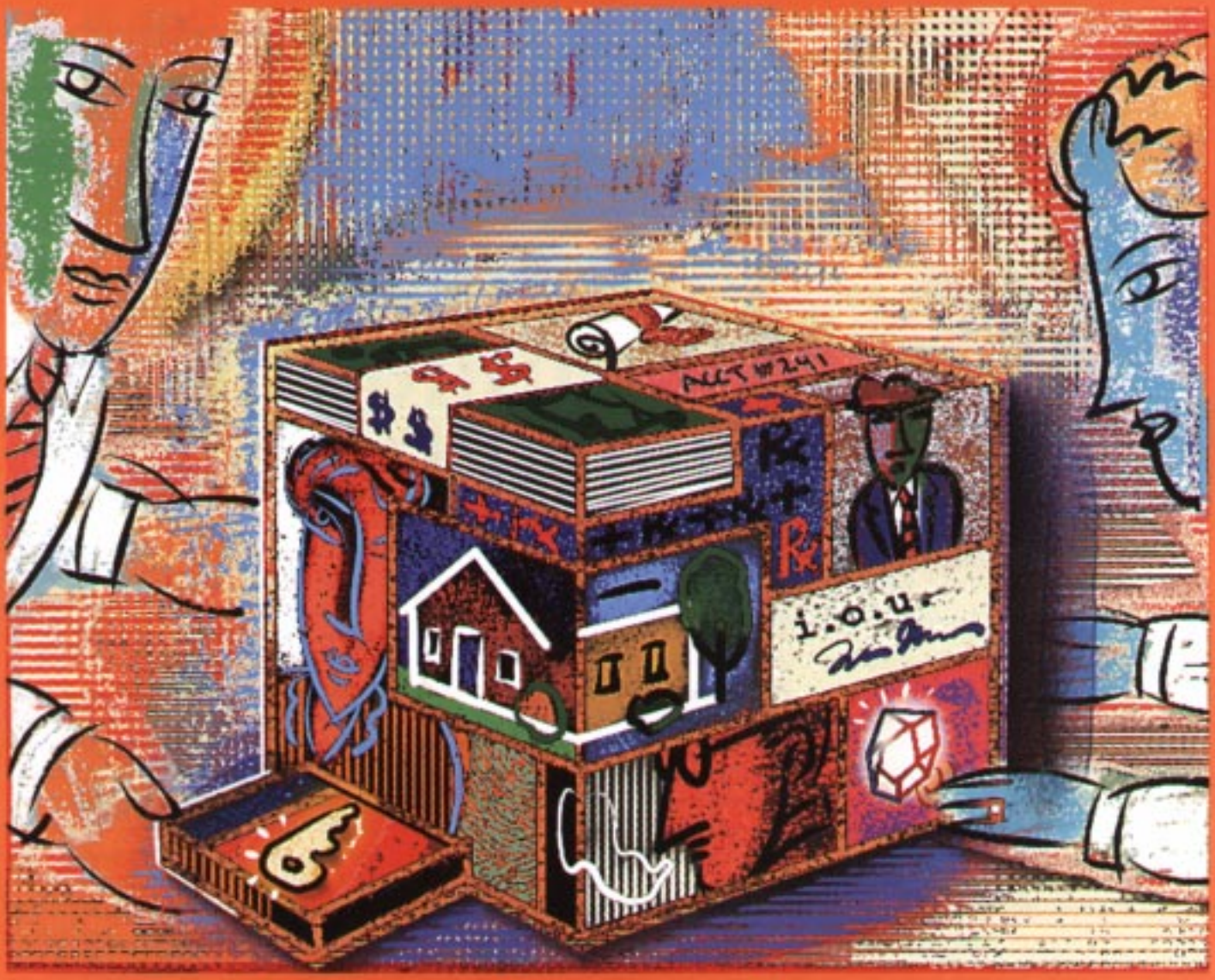


Designing Message BOXES

In a time when most online documentation specialists are focused on learning HTML-based help and are primarily concerned with knowing and using cutting-edge tools and technology, we may be forgetting one necessity for online documentation. While it is true that we need to learn efficient ways to provide online help and must master the tools that make us more productive, we must also remember that accomplishing these goals is secondary to offering software users fast, effective, and accurate online information.

By Kevin Lewis
Boston Chapter

This article discusses how to provide online information by using a source that many documentation specialists never work with—the message box. Many documentation specialists regard message boxes simply as part of the application and let software developers handle them. In reality, however, message boxes are where software documentation begins.

What Are Message Boxes?

The Microsoft® Manual of Style for Technical Publications defines a message box as "a secondary window that is displayed to inform a user about a particular condition." It stands to reason that the text you place inside a message box is called a message. In his book *Designing and Writing Online Documentation*, William Horton defines messages as "small pieces of information automatically displayed to guide users of a computer program."

A key characteristic of a message box is that it is *automatically displayed*. Automatically displaying information is far more effective than forcing users to search for it. Also, properly written messages *guide users*.

Messages do not necessarily have to be presented in small pieces. I like to think that every documentation specialist presents only the required amount of text in any type of documentation—online or hard copy. However, message boxes need not be limited to small pieces of information. They should include the appropriate amount of information to enable users to complete their jobs.

I am disappointed that none of the definitions I have found on message boxes mention that the *timing* of a message box is critical. Users not only need information fast but also may need it before performing another task.

Here is my own definition of a message box, as compounded from several sources. A message box is a window that a software application (not the user) displays to inform users about a condition, process, or procedure that the users should know about before continuing to use that application.

Why Should We Use Message Boxes?

I can tell you in one simple sentence why we should use message boxes—users need them. Message boxes give users specific information they need to perform a task or handle a system process. Users who are reluctant to fish around with an online help system can automatically get the information they need from message boxes.

As Cheryl Lockett Zubak pointed out at the 1998 WinWriters Online Help Conference, "Users are in a hurry...they don't want to wade around in your help system (no matter how cool it is)." She also pointed out that users want to *do things*—they prefer learning an application by using it. Since users are *doing*, we need to give them information while they are experimenting with an application.

Message boxes differ from online help in that users do not have to seek the information they need: The application automatically displays it for them. The best way to emphasize the significance of this automatic display of information is through an example.

A software developer recently asked me to help her write a message box for an application. Depending on the mode in which the software runs, users may need to know special information when exiting the application. Normally, however, users can exit without this information.

Automatically displaying information is far more effective than forcing users to search for it.

Also, properly written messages *guide users*.

The message box also eliminates the need to document the information in the reference manual and online help, since it is automatically displayed when users need it.

To document the information required for this possibility, a documentation specialist has two options. The first is to write out an explanation in the online help, detailing the circumstances under which the user must have the special information in order to exit the application. The users must then take the time to find and read the explanation. This option seems like a lot of work (for both the users and the documentation specialist), especially to explain something users do not usually have to know.

The more sensible option is to automatically display the information when users need it. In this example, we display a message box when users attempt to shut down the application. The message box gives them information they need about the task they are attempting to perform. In a sense, the message box makes the application self-documented. The message box also eliminates the need to document the information in the reference manual and online help, since it is automatically displayed when users need it.

Figure 1. Table Used for Message Box Drafts

ID	Message	Default Button	Other Button(s)	Icon

How to Construct Message Boxes

The first step in constructing message boxes is to determine where and when you need them. Generally, you should use a message box for errors, prerequisite actions, and confirmation of actions that could have serious implications to your users. Also, timing of a message box is crucial. Users must have fresh in their minds the actions they were taking before the message box appeared.

The next step is to gather information about the events that trigger the message boxes. What events cause the message boxes to open? How do these events affect the application and its users? What options, if any, do users have? What happens after users close the message boxes?

After you have gathered this information, it's time to start writing the messages. When you write a message, you should follow a certain structure to provide users with the following information:

- The event that occurred to generate the message box
- What will happen as a result of this event and if the user lets the event continue
- A summary question that leads to an answer by clicking a button

The summary question should be short but meaningful, because experienced users are likely to read only the question and ignore the rest of the message. Following the summary question, the message box should have buttons with labels that answer the question. The default button (the button that is activated if the user presses the Enter key) should be the answer users are most likely to select or the one that is least likely to have serious implications for system operations.

If your company uses icons in message boxes, be sure to use one that makes sense for the message. Three common icons are informational, critical, and warning. Use critical or warning icons only when the user's actions could have serious consequences.

You'll want to write the drafts of the message boxes in a word processor or spreadsheet application. Create a table similar to the one shown in Figure 1. Each row in the table represents one message box. The message box ID should correspond with the ID used in the application development tool. When you have finished writing and editing the message boxes, you can either give the table to the developer to update the message boxes or use it to update them yourself.

Figure 2. Poorly Designed Message Box

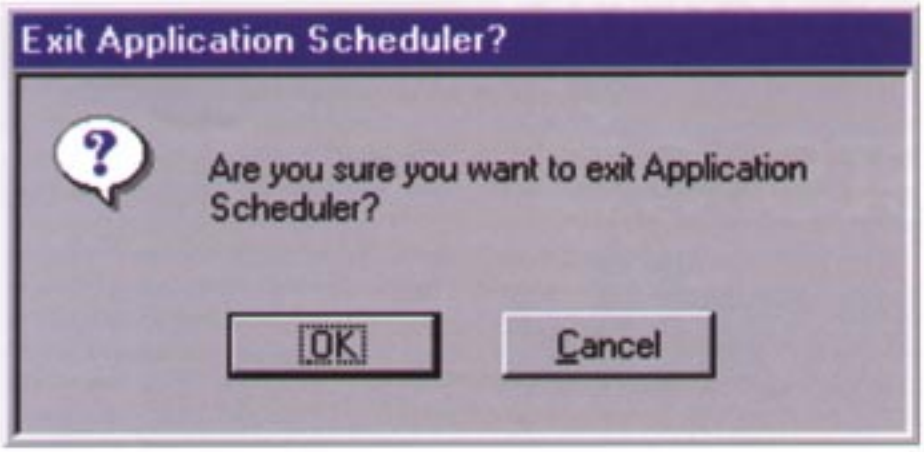
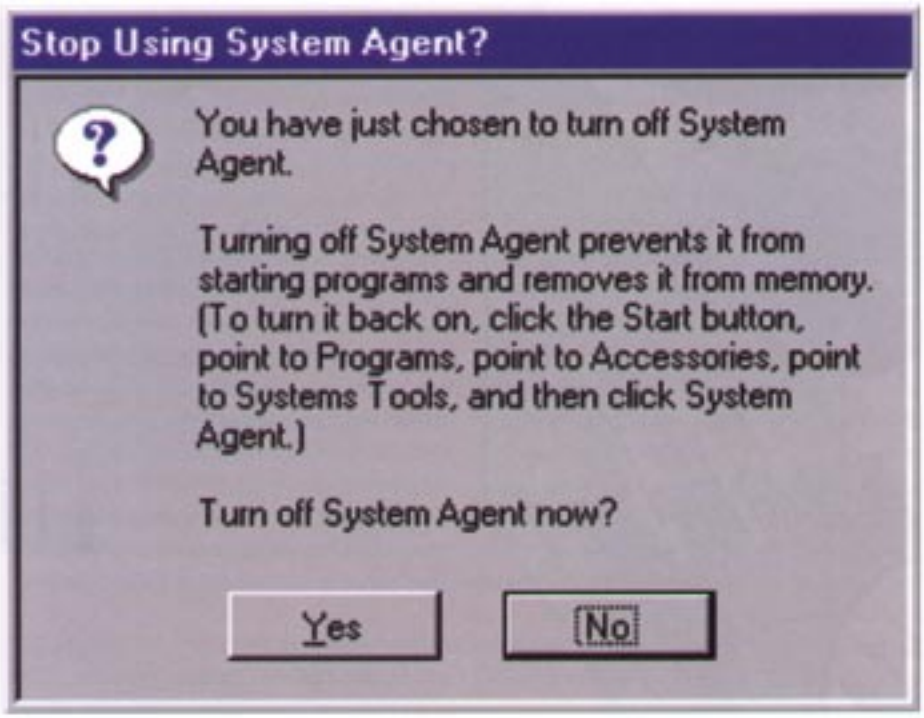


Figure 3. Well-Designed Message Box



Example

Let's use this information in an example. In this example we will document a message box for a fictitious program called Application Scheduler. Application Scheduler runs applications on the user's computer automatically at times scheduled by the user. Because of the nature of the program, shutting down Application Scheduler could significantly affect system operations. Scheduled applications cannot run when Application Scheduler is shut down.

Let's first look at how *not* to design the message box. Consider the message box in Figure 2.

This message box does not tell users what they did to generate the message box. If users think they clicked a "Print" button but accidentally clicked an "Exit" button, they will wonder why the "Print" button generated such a message box (of course there are considerable user interface design issues if an "Exit" button is mistaken for a "Print" button—but that's an article for another time).

The message box does not tell users anything about what is happening with the application, nor does it tell them what to expect if they let the event continue. Basically, the entire message is a summary question.

One final problem: The buttons in the message box do not answer the question the message box asks. If asked the question orally, users would most likely answer "Yes" or "No."

Let's compare this message box with one created for the same procedure in a similar application. *Microsoft® System Agent* offers a near-perfect example of a well-designed message box, shown in Figure 3.

Notice that the first sentence in the message tells users what event generated the message box. The next sentence tells what will happen if the event is allowed to continue. The message box ends with a summary question and buttons labeled with possible answers. "No" is the default answer, because answering "Yes" could significantly affect system operations.

Conclusion

Documentation specialists should treat message boxes as part of a documentation set. Instead of allowing random messages to vaguely describe system operations, we should use message boxes to help users work with the software application. While message boxes cannot replace traditional documentation, they can give users relevant online documentation when they need it. ■

SUGGESTED READING

Horton, William. *Designing and Writing Online Documentation.* (New York: John Wiley & Sons, 1994).

The Microsoft Manual of Style for Technical Publications. (Redmond, WA: Microsoft Press, 1995).

Zubak, Cheryl Lockett. *How Computer Users Seek Assistance.* (Symposium: Sixth Annual WinWriters Online Help Conference, 1998).

Kevin Lewis has a master's degree in technical and professional writing from Northeastern University. He currently works as a lead technical writer at IDX Systems Corporation in Boston. You can reach him at ***klewismail@aol.com.***