

# Introduction



Campbell Land/Stock Illustration Source

# to Single Source

Part 1  
BY PHILIP BUTLAND

*Editor's note: This article is the first of a pair that introduces single sourcing. Part 2 will run in a subsequent issue. For those ready to tackle a more advanced form of the same concept, see the article on page 28.*

Single sourcing has become a buzzword in documentation circles. No conference is complete without a couple of sessions on single sourcing. For a while, it was regarded as a fad that would likely disappear within a few months. But single sourcing is now flourishing, so I think it merits further discussion.

My experience from talking to people at my company, Agilent Technologies, is that many people are thinking about making a transition to single source. However, they're still not sure how painful this transition will be. I made the transition a couple of years ago. This article, drawing on my experience, presents the main advantages and problems involved with single sourcing.

## What Is Single Source?

I find it useful to use two definitions of single source. The first definition is the most widely used: to create manuals and help files from the same source documents (this is the sense in which the phrase "single source" is usually used in this article). However, we can use a broader definition: to use the same source document to produce multiple versions in any medium. This broad definition of single sourcing may be illustrated by the following examples:

- *Creating printed manuals and PDFs.* PDFs and manuals are often regarded as the same thing. They are not. They are read in different contexts, and on different media. In addition, every release of Adobe *Acrobat*, for example, enables new features for PDFs. And it is now commonplace to include hyperlinks or to jump to URLs in PDFs. We can no longer treat manuals and PDFs as being exactly the same thing.
- *Using the same source for pre- and post-sales literature.* This kind of single sourcing can reduce costs. For example Agilent Technologies produces technical data sheets, which contain the same information that is in the specifications appendix of a manual. Every so often, we get requests to translate this information. As it happens, all the information is translated, but only in manual form. If we were to access manuals and data sheets from the same database, we'd be able to provide translation for both at no extra cost.
- *Common information in a suite of products.* An example of this kind of single sourcing is a definition of terms that would be the same across related products.

In each of these cases, the information provided is the same, but the circumstances in which it is read are different. To create good single source documents, information must be written

and structured so that it makes sense in a number of different contexts. If a context-neutral language and presentation style are adopted, the same source information can be used, avoiding unnecessary duplication.

### Why Single Source?

The main arguments in support of single sourcing concern cost and consistency. Traditionally, writing online help in addition to manuals would double both the costs and the time needed to complete a job. Producing the help with single source uses minimal extra resources.

The savings are most marked when you consider localization. Obviously, if you are producing less written documentation, there is less to localize. However, you also save during updates. Most localizers use a system of “chunking” whereby they ignore all blocks of text that have not altered. The nature of single source production means that all your work is concentrated in a relatively small number of chunks, which makes your translators' jobs quicker and easier. At the WinWriters 2000 conference, a speaker estimated that on a typical, average-size localization project, single sourcing saved \$1 million.

The benefits are not just financial. If you're duplicating information (particularly figures or technical information that

you don't fully understand), there's always the possibility of errors and inconsistencies. If you use just one source for this information, you know that once the information is right, it's right everywhere. In addition, single-sourced information is easier to check, and you don't wear out the patience of research and development (or whomever) asking them to check the same information several times.

The concentration of single source documents means that documentation can be produced and updated much more quickly. Coupled with the quicker review time, this gives you the chance to meet those impossible deadlines.

### Why Not Single Source?

There are two main objections to single sourcing. Both are legitimate, as far as they go.

The first objection is that manuals and help are not the same: they are read in different circumstances and by different people, and text that looks good in a manual makes no sense when it appears on a help screen. These points are certainly true, and I will elaborate on them later. However, if you write in a structured, medium-neutral style, you can produce text that is acceptable in all media.

The second objection is that single source writing requires extra work. This is also true, although the amount of initial work is usually offset by the time and effort saved later. It is important to remember that these later benefits are gained only on genuine single source projects. Sarah O'Keefe—president of Scriptorium Publishing Services, Inc., and a consultant who has worked on a variety of single sourcing projects—esti-



The main arguments in support of single sourcing concern cost and consistency.

The adage  
 “No one ever  
 reads manuals”  
 is not **entirely** true.



mates that a single source project is worthwhile only if there is at least 50 percent common text between the manuals and the help. Therefore, if most of the information in your manual should not appear in help, single sourcing is not worth the effort. On the other hand, if you need to present roughly the same information in manuals and help, the savings can be enormous.

### The Differences between Manuals and Help

The biggest argument against single sourcing maintains that, because manuals and help are different, text written for a manual does not sit well in help, and vice versa. This argument is generally correct. It also helps explain the limitations of some single source solutions, such as trying to generate help by pasting pages out of a manual or trying to generate a manual by sticking an introduction onto a series of help pages. These solutions do not address the fact that the structure of manuals and help is intrinsically different.

If you are aware of the differences between manuals and help (in intended audience, in structure, and so on), you can avoid the words and phrases that sound so clunky in a different medium. It's not a perfect solution, but it's near enough perfection to justify the savings you'll make elsewhere.

### Manuals

Manuals are read in a linear sequence. Although readers may dip in, they usually start at the beginning and go toward the end. It is therefore reasonable to assume knowledge of concepts introduced earlier in a chapter or section. Because a manual is a physical object, the reader can see both a left and a right page. Discrete chunks of information can be presented in up to two pages, unlike help and PDF files. Cross-references are helpful, but only if they are used sparingly. If a page contains dozens of cross-references, the reader will not be able to follow the flow of the description.

The limitation on a manual's size depends on its binding and on the context in which it is being used. Manuals used in the field need to be portable. Users may not want yet another large manual taking up bookshelf space.

Also, a manual is independent of what it is describing. It is not part of the software, and when you reach a certain part of a procedure, the manual does not automatically fall open at the correct page.

Finally, although it's not quite true that no one ever reads manuals, you cannot assume that just because you've written something everyone will read it. Manuals are consulted in particular circumstances, usually to obtain introductory or reference information. But many users

do not want to break from a task to try to remember where they last put their manual. An engineer whom I recently spoke with said he never takes a manual with him when he is visiting a customer. He'd rather give an incomplete answer than wade through a 300-page book.

### Help

Help is not read in a particular order. It is possible for users to read help from beginning to end, but it is far more likely that they have accessed the help in context (via the F1 key or help button) or from some sort of search index. Therefore, you cannot assume that the reader has read anything other than the information on the current screen. Because of the nonlinear nature of the information, expressions such as “chapter” or “the previous section” do not make sense, and section or figure numbers have no referents. Help is context-sensitive, so some assumptions can be made about what the user can see. But take care: the user may have used the navigation or search capabilities to access a particular screen.

Another feature which differentiates help from manuals is that each help screen is connected to other screens by hyperlinks. A user may quickly move to another screen and back again via the Back and Forward buttons. This feature allows you to provide more detailed infor-

mation or sample procedures without breaking the flow of your text. However, if text contains too many hyperlinks, readers can lose their sense of flow, and the text becomes more difficult to follow.

Help is usually the users' first port of call in a crisis. They want a solution to a particular problem, and the fact that they're using help means that they may not be in the best humor. The main purpose of help is to provide the information that is required with a minimum of mouse clicks.

### Source File for a Single Source Document

Single source text should be medium-neutral: The language and presentation must be appropriate for all media. However, manuals and help are structured differently, and you must find a way of organizing your source documents to reflect this fact.

There are a number of different ways to address this problem. Some systems, such as *RoboHelp*, start with a help document and create a manual by pasting the help pages together. This process adds structure—sometimes effectively, sometimes less so.

Other systems expect you to create help from a manual, as if ripping pages out of a book. For example, the documentation for *DocToHelp* says, "As long as you have *Word* and *DocToHelp*, you can convert almost any manual to help." In my opinion, this is starting in the wrong place: you do not want to create help pages from a manual (or vice versa), but to create a source which reflects the best qualities of each medium. Because manuals, unlike help, already contain a linear structure, the order in which items appear matters much more than in help. Each piece of information comes in the context of what comes before and after. For this reason, it is easier to break a manual into smaller segments. Each segment must be discrete to allow for the generation of help screens, but it is better to start with a structure than to impose one later. If you apply some structure to your manuals *before* you write them, you should be able to use them as the source for your help.

In my opinion, the most effective way



## The lowest common denominator for both manuals and help can be called a segment.

of producing single source software is to use *FrameMaker* and *WebWorks Publisher*. The basic structure of a *FrameMaker* document is maintained, with a *FrameMaker* book file as the source document. You will have separate book files for the manual and help because the front matter is different, but the chapters will be essentially the same. Chapters are divided into sections, subsections, and so on. Each page of help is equivalent to one of the smaller subsections within the manual.

### Writing in Segments

The lowest common denominator for both manuals and help can be called a segment. This is equivalent to a single help screen, or a section within a manual. It is usually a few paragraphs long (between half a page and a page). Segments must not be too long or too short. Size is particularly important for help screens, where a sentence of information is of little use on its own and users don't

want to scroll through several pages before they find what they need. Making sure that your segments are the right size requires some forethought. Single source software is smart, but not that smart.

Generally, the software creates a new segment whenever there is a header of a certain level (say, at every chapter header, section header, and subsection header). So if you immediately follow a chapter header with a section header, you will produce an empty segment, which can be acceptable in manuals, but not in help. Be aware that the text you use will now appear in both manuals and help, so you should be especially careful about the terminology you use.

### Section Header Names

The single source software uses the header name at the beginning of a segment as the title of a help page. This title will be used in the Navigation and Search facilities of the help. You should bear this in mind when choosing the text for your header. In a manual, it may be acceptable to begin each chapter with a section headed *Introduction*. The reader has other, contextual information to understand what is being introduced. But a help search that brings up twenty screens with the same header is worthless.

Figure 1 shows an example of such a help page. Each chapter in the manual has an Overview section. When this is translated to the help, the help index becomes unintelligible. It is therefore preferable to call your headers "Introduction to *whatever*," "Overview of *whichever*," or something similar. Always provide enough information so readers know what will be in this segment before they read anything inside.

### Conditional Text

Conditional Text is text that appears only in either the printed or online document. It is used when information is necessarily different between the two media. In my opinion, it should be used as sparingly as possible. It is far easier to maintain a document if you choose a vocabulary acceptable to all media. However, conditional text is sometimes necessary, as in the following examples:

- *Description of text that looks different in*

print than it looks online. Your character format *strong*, for example, may appear bolded in print and italicized online. Any text that describes this difference will have to be different in each medium.

- *Information that does not fit in the other medium.* A manual may contain more conceptual information than the help, or there may be more room in the help for examples. I would advise leaving out entire chapters wherever possible, which will eliminate the need to use conditional text at all. Where this is not possible, try to mark as conditional a large chunk of text (a whole paragraph, at least). Localizers will appreciate your effort.
- *Introductory text.* A procedure may have an opening sentence in help that is obvious from the context of the manual. Again, try to mark the entire paragraph.
- *Cross-references and links.* With powerful single source software, links in manuals and help will be automatically interpreted in different ways. All you have to do is add a cross-reference using normal *FrameMaker* functions.

## Terminology

You may be wondering what terminology you should use in your source document. It is easier to say which terminology you should not use. Single source text should be medium-neutral. In other words, you should not use any terms that are used only in manuals or only in help. Do not use terms that do the following:

- Refer to the structure of a manual; for example *See chapter ABC* or in the *Appendix*.
- Assume an order in which sections are read, such as *in the preceding/following section, As we have seen, (above), and Next (at the start of a section)*. However, if you are referring to text within the same section, this sort of vocabulary is perfectly valid.
- Refer to numbering not used on a help screen; for example, *Figure 4.2, Section 2.3*.
- Assume the user can see the software (when you are reading a manual, the software is not necessarily running); for example, *Click here to...* Instead, you should make explicit references, such as *Click the OK button*.


## Conclusion

In my experience, single source is worth the effort. But we need to understand that single source is more than just a fancy software package. At least as important as using the software is making sure that your document is written and structured in a way that facilitates single sourcing.

I have encountered some intense hostility to single sourcing. Most of this hostility is understandable when single sourcing is imposed from the outside. However, once your single source system is in place, the savings will be noticeable. It has often taken me much less time to write a single source document from scratch than to rework an old document to give it proper structure.

Single sourcing is not the solution to all your problems. It will not make all your projects quicker or cheaper, and you need to be able to decide when it will be useful. This is why all new technical writers in our department participate in a one-day course on the theory of documentation structure. The course may sound abstract and unnecessary, but in the long run, it ensures that we are able to use single source with proper forethought.

Finally, I would like to quote Sarah O'Keefe once more: "Single sourcing works only when you create highly structured content and then express that structure in different media to take advantage of each medium's strengths." With proper structure, and an overall view of what you are doing, single sourcing can be the solution that you didn't know you were looking for.

My next article will include some practical examples of how I structure my *FrameMaker* documents to produce the source documents that I need for single sourcing. 

---

*Philip Butland is a learning products engineer at Agilent Technologies in Germany. He is a member of Agilent's Learning Products Council and is currently working on the design team for Agilent's manuals and help files.*

Figure 1. Help file with unspecific headers

