Introduction

This book will help you negotiate, draft, and understand information technology contracts. Specifically, it will help you address software-as-a-service and other cloud computing agreements, on-premise software licenses and other software transfers, and technology professional services agreements. It will also help you with the many contracts that combine two or more of those offerings. This book addresses business-to-business contracts (between companies) and business-to-government contracts. That includes agreements related to technology outsourcing and software distribution, among many other types. This book also addresses business-to-consumer contracts.

The guidance offered here is for both lawyers and nonlawyers. The text stays away from technical jargon—"legalese," "engineerese," and "programmerese"—and where it absolutely can't avoid jargon, it provides a definition. In other words, this book uses simple English, like a good contract.

You can use this book as a training manual or a reference guide or both. If you're training, read this book cover to cover. It provides an overview of the key technology contracting concepts.

If you're after a reference guide, you can pick and choose the chapters to read. When you're negotiating a contract, or reading or writing one, look up the various clauses to learn what they mean and what's at stake. You'll find sample language in each chapter, which you can incorporate into your own contracts. Plus, if you visit this book's website, http://TechContracts.com, you can copy the sample clauses and paste them into your document. You'll also find full-length contracts at the website, which you can download and revise to fit your deals.

Finally, you can use this book's table of contents as an issue spotter: a checklist of clauses to consider.

This book can't replace a lawyer—or a colleague with more information technology (IT) experience, if you are a lawyer. But it can help you understand your lawyer or colleague. And whether you have legal help or not, the better you understand your contracts, the more effective you'll be.

I'm a technology lawyer and expert witness, a law school lecturer, and a professional trainer. This book grew out of seminars I teach, for both attorneys and non-attorneys. When I started running those seminars, students often asked where they could learn more—if I knew a good book on IT contracts. Most of the books I knew were massive tomes on intellectual property or contract law. They're written for lawyers only, and their more practical lessons are spread across hundreds of pages. I've learned much of my trade on the job, rather than from a book. I've served as a technology lawyer in San Francisco and the Silicon Valley, both with a global firm and with my own law firm. I've also served as general counsel for a publicly traded software company and as vice president of business development for an Internet start-up. The material for my seminars came from the contracts I had negotiated and written in those positions. I'd never seen a really user-friendly outline of the issues. So I wrote this book.

This is the third edition. I've revised each edition to address changes in law and industry practices, as well as my own continuing experience, particularly as a contract negotiator and expert witness. And, in this edition, I've added something new. I've explained a handful of the most complicated issues raised by typical IT contracts. Those include the more intricate twists and turns of indemnities, the confusing role of "cumulative" limits of liability, and the realities that debunk myths surrounding copyleft ("viral") open source software.[1] Many contract drafters get along without ever noticing these trickier issues. And some who notice them ignore them—and that can be a valid choice. Figuring out, drafting, and negotiating *any* contract term costs time and money, and you should always weigh those costs against the benefits. So, in this book, each really complex issue has

1. See Subchapter II.L.5 ("The Cause Problem (or Advanced Indemnities)"), Subchapter II.M.1 ("Dollar Cap"), and the section called "Why Copyleft Probably Isn't 'Viral'" in Appendix 2 ("Open Source Software Licenses").

its own subchapter or section (the latter separated out by three diamonds, like those beneath this paragraph). That way, you can easily skip an issue you don't want to tackle without skipping more vital, everyday explanations. But I hope you will dive into all the issues addressed here, including the handful of unusually complicated ones. Even if you decide not to address an issue in your contract, you'll do a better job if you're aware of it and understand it.

◆ ◆ ◆ ◆

The rest of this introduction provides more detail about the types of contracts this book covers. It also explains the structure of a contract and of this book, and it offers a few explanations that will help you get the most out of your reading. Finally, it provides a short explanation of some IT industry language—just a little, particularly regarding cloud computing—and then offers three lessons about contracting in general.

Subject Matter: Types of IT Agreements

This book addresses four principal types of IT contracts, along with a fifth type that blends the others:

1. **On-Premise Software License Agreements** and **Distribution Software License Agreements:** The provider gives the customer the right to copy software. And sometimes the provider authorizes distribution of software or the exercise of other rights under intellectual property law, particularly under copyright. These contracts include end-user license agreements (EULAs), where the customer gets the right to copy the software onto its own computers ("on-premise"). They also include distribution agreements, where the customer—more accurately called the "distributor"—gets the right to circulate copies to its own customers. In each case, the provider keeps ownership of the software.
2. **IT Professional Services Agreements:** The provider promises to use its people to help the customer. It promises consulting or software programming or tech support or some other assistance a human professional can provide.

3. **Cloud Services Agreements:** The provider promises to give the customer remote access to its computers and software. The customer doesn't get possession of the computers or a copy of the software—just access. Cloud services fall into three primary categories: software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS)—all described under "A Little Industry Language, Particularly re Cloud Services," later in this Introduction.
4. **Software Ownership Transfers:** The provider transfers ownership of software copyrights to the customer—or sometimes ownership of patents or intellectual property (IP) rights. These deals resemble on-premise software licenses (see #1), because both involve grants of IP rights. But here the customer gets ownership of the IP, while in license agreements, the provider retains ownership and the customer just gets limited rights.
5. **Combination Agreements:** The provider gives the customer some combination of the four above. For instance, the provider might provide SaaS or another cloud service (see #3), along with tech support as a professional service (see #2), to help the customer use the technology.

This book will also help you with purchase and lease agreements for computers and other IT hardware. That's because many of the clauses discussed here appear in hardware contracts too, and some of the contracts listed here can include hardware purchases. But hardware agreements involve some terms you don't see in other types of IT contracts, like equipment leases, security interests, and shipping terms. This book doesn't cover those clauses.

Finally, this book addresses government contracts, though with one limit. A government contract is an agreement between a provider and a government agency, as the customer. It can involve any of the contract types just described, and, if so, this book will help you. But some government contracts include language required by government contracting laws and rules, like the federal acquisition regulations (FARs) and state procurement regulations. This book doesn't address those rules, though it should still help you understand much of the language they require.

The Structure of a Contract and of This Book

IT contract terms can be organized into three groups: *prime clauses*, *general clauses*, and *boilerplate clauses*. This book is organized the same way.[2]

The *prime clauses* express the deal's central terms. There, the provider grants a license or other rights to on-premise software or to software meant for distribution. Or it promises to provide professional services or cloud services. Or it transfers ownership of software. Or it does some combination of those four, in a combination contract. The customer, on the other hand, promises to pay. This book addresses prime clauses in Part I.

The *general clauses* account for most of the contract. They cover everything not addressed in the prime clauses or the boilerplate clauses, and they're usually the most heavily negotiated. This book addresses the general clauses in Part II.

Boilerplate clauses cover the theoretically noncontroversial mechanics of a deal: terms on independent contractor status, contract interpretation, choice of law, etc. IT professionals tend to put most of these terms at the end of a contract. This book addresses boilerplate clauses in Part III.

Contracts usually start with two sets of boilerplate clauses: the introduction (including "recitals") and the definitions.[3] From there on, you should organize your clauses the way they're listed here: prime clauses, then general clauses, then the remaining boilerplate clauses. That makes agreements easy to understand. Unfortunately, though, you'll probably run across contracts with these clause types jumbled together.

---

2. You might not find these terms outside this book (the same goes for "combination contract" above), but they're meant to express common views of these clause types.

3. See Chapters III.A ("Introduction and Recitals") and III.B ("Definitions").

Using This Book

The following four brief notes and explanations will help you get the most out of this book.

First, almost any contract you write should be customized to fit your deal. So if you insert one of this book's sample clauses—which is what they're here for—don't do it thoughtlessly. You may need to edit it. Think through the unique issues raised by your deal. For example, "We know the Windows version of our software has some serious bugs, so we don't want to give a broad warranty." "We've got a small IT department, so we need extra support." "Our CFO gets hives if we give data incident indemnities." This book offers building blocks for a contract addressing issues like those, but the customizations are up to you.

Second, this book primarily addresses U.S. law. That doesn't mean you shouldn't use it for contracts under other countries' laws. Most contract clauses mean what they say, regardless of the underlying legal system. But with some clauses, the underlying law really will affect the meaning. It's always a good idea to get help from a lawyer who knows the relevant jurisdiction, and that's all the more so when you use this book for deals outside the United States.

Within the United States, the 50 states have similar contract laws, and federal law governs some of the issues discussed here, particularly related to copyrights and patents. So state law variations won't often lead your IT contracts astray. But some state law variations really do matter. That's another reason to consider help from a lawyer with relevant experience.

Third, like most contracts, the examples in this book use defined terms. When a contract creates a concept and uses it more than once, it usually defines it. For instance, a contract might list the provider's services in Section 2, then mention them over and over in other sections. Rather than listing the services repeatedly, the contract defines the list as the "Services." Whenever the contract refers to the "Services" with a capital S, it means the whole list. This book's sample clauses work the same way. (Some contracts mark defined terms with all caps instead—e.g., the "SERVICES.") The same goes for sets of initials in all caps, like "NDA" (for nondisclosure agreement). In this book, some sample clauses don't supply the definition. That's because, in a contract, another section would define that term. Obviously, in your contracts, you should provide the definitions somewhere.

Fourth, most of this book's sample clauses use the defined terms "Provider" and "Customer." The text uses them too, though without the caps. "Provider" stands in for "Vendor," "Licensor," "Supplier," "Seller," "Consultant," and "Service Provider," among other names contracts use for the selling party. And "Customer" stands in for "Licensee," "Buyer," and "Client," among others. This book favors "Provider" and "Customer" because they're generic. But the text and sample clauses also occasionally use other names like "Distributor" and "Discloser," where they fit the topic.

A Little Industry Language, Particularly Re Cloud Services

This book includes the world's shortest glossary, at the end. It explains five terms you'll see in the text: *calendar* (as in "*calendar quarter*"), *including without limitation*, *object code*, *source code*, and *without limiting the generality of the foregoing*.

This book also uses some terms related to cloud computing, and they're important enough to explain up front. The key terms are "cloud services" and "cloud computing services," as well as "software-as-a-service," "platform-as-a-service," "infrastructure-as-a-service," and "cloud computing" itself.

"**Cloud services**" and "**cloud computing services**" mean the same thing. In this business model, the provider hosts software or other technology and the customer accesses it remotely, almost always via the Internet.

Many professionals confuse cloud services with software licensing. As you'll see in Chapter I.D ("Subscription for Cloud Services"), cloud services don't involve licenses to software. (At least, they shouldn't.) Licenses are for on-premise software agreements, where the customer makes copies of the software to run *on its premises*—and so needs a copyright license. In cloud services, the provider keeps the software to itself; it doesn't give the customer any copies. Rather, the provider gives the customer access to its technology: the chance to use it (often called a "subscription," rather than a "license").

Cloud services are often confused with *professional* services too. Again, that's a mistake. The cloud services customer does not get services from the provider's *people*: its professionals. Rather, services come from the provider's *computers and software*. (The term "cloud

services" generates much of the confusion since "services" suggests help from *people*. It's a shame the industry didn't settle on "cloud access" or something like that, but that train has sailed.)

The best-known type of cloud service is **software-as-a-service** (**SaaS**). In a SaaS deal, the provider gives the customer remote access to a software *application*. An application is software for end-users, like a word processing or contacts management program.

IT professionals typically talk about two other types of cloud services. In a "**platform-as-a-service**" (**PaaS**) offering, the provider gives the customer remote access to a software *platform*. Platforms are not for end-users. They're for developers, who build software "on top of" platforms. (Think of the platform as a tabletop workbench full of tools.) Often, those developers also run their software applications on top of the platform. Finally, in an **infrastructure-as-a-service** (**IaaS**) offering, the provider gives the customer remote access to computers and other hardware infrastructure. The customer installs software applications and maybe platforms on that infrastructure and uses them.

Finally, the term "**cloud computing**" can generate some confusion too. As explained earlier, this book uses "cloud computing services" and "cloud services" for a *business model*, where the provider hosts technology for the customer. But without the word "services," "cloud computing" generally serves as a *technical term*. It refers to a type of technology architecture: an arrangement of technology. There, the key software operates on central server computers, and other computers access it remotely. Unlike cloud services as a business model, the cloud computing technical term doesn't focus on who hosts the computers or software. A customer could operate a system with cloud computing architecture all by itself, or it could hire an IT provider to host some of it. You'll avoid confusion if you remember that the technical term, "cloud computing," and the business model refer to different things. This book mostly discusses the latter: the "cloud *services*" business model.[4]

4. If the customer hosts the whole system, it's usually called, "customer-hosted cloud computing" or a "customer-hosted private cloud." This book doesn't talk much about those arrangements, but that's not because they're unimportant. Rather, if the customer hosts the cloud computing software itself, its purchase contract involves on-premise software. So this book addresses it in our broader discussion of on-premise software deals.

Three Lessons about Contracting

1. Good Fences Make Good Neighbors

Why do we sign contracts? It's not because we want to win a lawsuit later. It's not because we don't trust each other. It's not even because we're afraid lawyers will stir up revolution if they're not kept busy.

We sign contracts because good fences make good neighbors.

The best way to avoid arguments in a business relationship is to write down the parties' expectations ahead of time. That list becomes a boundary marker—like a fence between neighboring yards—explaining who's responsible for what. If the parties disagree, they can look at the list for guidance.

Contracts, then, *prevent* disputes—at least, good ones do. They prevent lawsuits.

Even if the parties never look back at the contract after they sign, it's still probably played a vital role. When people put their business expectations on paper, they often find those expectations don't match. Just the act of negotiating a contract will uncover many mismatched expectations. The parties can address them before starting work.

Yes, it's true that we sometimes sue over contracts. And yes, in interpreting a contract, we often talk about what a judge would say it means. But that's only because courts have the ultimate say if the parties can't agree. Job number one for the contract is to keep the parties out of court.

2. There Is No Such Thing as "Legalese" or "Technicalese"

You may feel uncomfortable with contracts because of the unfamiliar language they use. Don't be intimidated. You can understand most contracts.

There really is no such thing as *legalese*. Contracts are written in English or Spanish or Vietnamese or whatever language the parties speak, not in some special legal language. But contracts do sometimes use special shorthand: terms lawyers have developed to save time. And some IT contracts use technology-related shorthand. Finally, contracts sometimes use formal, stilted language with long run-on sentences. Don't let shorthand or stilted language bother you.

If you run into an unfamiliar term in a contract, don't worry. Look it up. You can probably find the definition online. And if it's legal shorthand, there's a good chance you'll find it in *Black's Law Dictionary*,[5] found in many libraries. Or ask someone with the right expertise.

Once you understand a term, feel free to use it in your own contracts. But you should also feel free *not* to use it. Shorthand is optional. If you do use shorthand, be sure the contract defines each technical term. Definitions can vary for IT terms like "sandboxing" and "bot,"[6] so the contract needs an agreed definition, unless there really can't be any doubt. Legal terms, on the other hand, often have widely accepted definitions, so you don't necessarily need to define them in the contract. But if in doubt, define.

As for long sentences, just take a deep breath and read slowly. The same goes for formal language. There really is no reason to use terms like "heretofore" and "*lex loci*."[7] That sort of language often appears in form contracts from the olden days, when formal writing and Latin seemed more appropriate. It does crop up in modern contracts, often because someone wants to show off a big vocabulary. Be suitably impressed. Then take out your dictionary if necessary and figure out what each sentence says.

3. Ask Yourself "What's Our Best Option?" Not "What's Fair?"

Some businesspeople and lawyers ponder and argue a lot about whether proposed contract terms would be *fair*. I think that's an unhelpful view of contracts, for two reasons. First, it's hard to define "fair" in contract negotiations. Second, a focus on what's fair may lead you to reject deals that make economic sense, or to accept deals that don't. The better question is: would doing the deal under these terms be more profitable than *not* doing it?

What does "fair" even mean in contract negotiations? Each party has a choice about whether to do a deal, so neither owes the other any

5. From the West Publishing Company.

6. *Sandboxing*: separating a computer program from other programs to limit the impact of errors and security issues. *Bot*: a software program that mimics human behavior in that it's automated (short for *robot*).

7. *Heretofore:* before now. *Lex loci:* Latin for the law of the place, usually referring to a contract's choice-of-law clause.

particular terms. If some company insists on terms heavily slanted in its favor, and no one ever accepts them, that company's dumb, not unfair. Or maybe it just doesn't care whether it does any deals. On the other hand, if enough people do accept the bad terms, does it make sense to call them unfair? Does it even make sense to call them *bad terms*? The fact that "the market" accepts the terms legitimates them.[8]

So you might say "fair" can't be defined in contract negotiations. Or you might say "fair" means "acceptable to enough people." Either way, the guiding principle behind contract terms is leverage: whether the proposing party can get its terms often enough to do the deals it needs.

If you do focus on what's fair, you might walk away from deals that make economic sense. If terms you don't like seem unfair, and you can't get the other party to budge, you'll probably feel too screwed to sign the contract. But that's a poor choice if you couldn't get better terms from anyone else, and if accepting the terms would be more profitable than dropping the project. What if you focus on *best option available*, rather than fairness, and recognize that the other party wants its best option too? In that case, you'll accept the deal if it's the best option—and you'll feel good about it. You'll only walk away if the market offers better options.

A fairness focus could cut the other way too. You might make concessions because the other side's requests sound fair—even though you've got options better than doing the deal under those terms. If you simply ask yourself whether you've got better options, and the answer is *yes*, you'll refuse the other side's "fair" terms.

I'm not suggesting contract negotiators should be androids or Vulcans, who react only to logic. Most of us want to leave the bargaining table happy—and if we're smart, we want the other side to feel that way too. And most of us *do* respond to arguments about fairness. We just need to remember how slippery the concept is, and how ultimately the market for other options shapes the definition of "fair," rather than some objective concept of right and wrong.

---

8. OK, that's not entirely true, at least so far as the law is concerned. Courts won't enforce certain heavily slanted contract terms because they're "opposed to public policy" or "unconscionable." See Subchapter II.M.3 ("Unconscionability and Required Clarifications").