VistaLaw

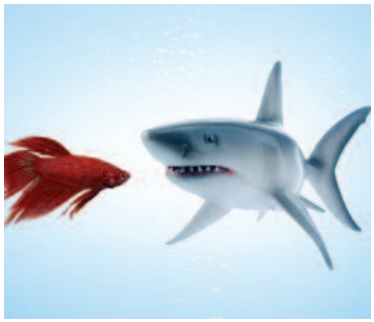
Views

Drafting and Negotiating SaaS Agreements: Traps for the Unwary

March 2009

Michael L. Whitener, VistaLaw International LLC

In this issue: *As the SaaS industry grows by leaps and bounds, the principles of software contract drafting and negotiating have to change to keep pace with SaaS developments. Standard software license templates are not up to the job. This article addresses the most common pitfalls of SaaS agreements. The good news is that these pitfalls are avoidable with a bit of upfront awareness, clear and concise contracts and an understanding that SaaS transactions require a new approach.*



The SaaS transaction is a unique animal, and requires its own specific terms and conditions.

With the software as a service (SaaS) model expected to account for up to 50 percent of the entire software market by the year 2013, SaaS certainly qualifies as the "new new thing." But underlying every SaaS deal is a legal contract between vendor and customer – and to judge from the SaaS agreements crossing my desk, they're still largely based on the "old old" software license template.

This won't do. A standard software end user license agreement (EULA) can't simply be given a nip here, a tuck there, and then be expected to perform the work of a properly tailored SaaS agreement. The SaaS transaction is a unique animal, and requires its own specific terms and conditions.

From the vendor's perspective, here are nine major pitfalls of SaaS agreements to be aware of and avoid:

1. Failure to address data security concerns.

A major distinction between a standard software license arrangement and a SaaS platform is where the customer data resides. When a customer enters into a SaaS agreement, the customer is putting its precious, sensitive data into the hands of the SaaS provider and its off-site hosting capabilities. A customer with any sophistication is going to demand assurances that its data is secure, and a smart SaaS company will anticipate and satisfy those demands.

The SaaS agreement therefore should address, at a minimum, the following concerns:

- Ownership of data
- Safeguarding of data, including keeping it from prying eyes
- Preventing the commingling of the data of various customers
- Return of data upon termination of the agreement or upon demand
- Recovery of data in the event of system failure
- Fate of data if the software vendor disappears

The SaaS vendor should be wary of over-promising in the area of data security, because the potential liability risk for data security breaches may be huge, especially when personal privacy rights are violated. One sensible approach is to simply describe what security measures the vendor takes to protect data, without implying that those measures provide ironclad protection. Warranty and indemnity provisions should be designed to minimize the vendor's exposure to lawsuits for inadvertent data releases.

The SaaS vendor must become familiar with various certification programs that dictate well-documented security practices to govern their data center operations and personnel. For example, if a customer is a public company, the Sarbanes-Oxley Act of 2002 dictates that the customer demonstrate and maintain effective corporate controls across several business functions. Therefore the customer may require that the SaaS vendor receive a Statement on Auditing Standards (SAS) 70 Type II audit certification, aimed specifically at service organizations, which ensures that the vendor has appropriate controls in place to meet the customer's audit requirements. In particular industries, such as health and finance, the SaaS vendor may be required to satisfy highly detailed data privacy requirements in order to pass muster.

2. Sloppy Service Level Agreements.

The Service Level Agreement (SLA) sets the customer's expectations regarding such critical variables as uptime and response time, and provides a remedy if the promised service levels are not met. Therefore it's an aspect of the SaaS offering that deserves to be crafted with great care.

Surprisingly, however, this is rarely done. For instance, a typical SLA may define promised uptime in terms of a percentage of time that the service is operational and available (99.999%, or "five 9s," being the Holy Grail of service level assurance for truly business-critical applications, while "three 9s" are more typical). But are scheduled maintenance and system upgrade periods considered as downtime by this definition? Without an explicit exception, they probably are. Is the percentage calculated on a daily, weekly or monthly basis? Who keeps the service records, and are they subject to audit? The agreement is often silent on these issues, which may lead the sophisticated customer to doubt the robustness of the service level promises.

It's extremely important that the agreement specify that credits are the only ("sole and exclusive") remedy for failure to meet service level promises.

Similarly, while response time is usually defined in terms of how quickly the vendor will react once notified of a problem (with required response times usually varying based upon the severity of the problem – critical, urgent or low), SaaS agreements are often ambiguous as to what constitutes a "response." At one extreme, it means the problem has been resolved; at the other, it only means that the problem has been logged into the vendor's system. This kind of ambiguity can only lead to confusion and resentment between vendor and customer if problems arise.

Finally, there's the question of what remedy is available to the customer in the event that service levels are not met. The usual remedy is to give the customer a "credit" based upon the percentage of time that the system was down. But it's shocking how many agreements fail to describe exactly what those credits are – whether a rebate or an extension of the service period – and the mechanics of the reimbursement. If there is a maximum credit limit for a particular period, the SLA must make that clear.

Also, it's extremely important, from the vendor's perspective, that the agreement specify that credits are the only ("sole and exclusive") remedy for failure to meet service level promises. Otherwise, a vendor may find itself facing a damages claim from a customer that decides that credits alone are not adequate to compensate for the lost service.

3. Lack of pricing flexibility.

The traditional software licensing model is simple: the customer pays up-front for the software, and the license to use the software is perpetual, absent a breach of the EULA. There is no option for customer return of the software, unless there is a material failure of the software to meet its specifications. If the software fails to live up to the customer's expectations – as opposed to meeting the written specs – tough luck for the customer.

That approach doesn't work with SaaS. Customers often expect to be given a chance to use the software at no risk. Free trials are common, and provide a low-cost way to offer the potential customer immediate gratification as well as create an opportunity to convert that lead into a permanent customer. A SaaS agreement can provide that payment is due only after the free trial period has elapsed without the customer terminating the service. The agreement can also give customers various options for frequency of payment and length of committed term for the service, with discounts for bigger payments in advance and longer terms.

Once a customer has committed to a specific term, it may still want the flexibility to expand or reduce the number of users of the service. Assuming that payment terms are based upon the number of authorized users, pricing should follow, perhaps based upon specified authorized user bands.

But defining "authorized user" can be a challenge. Suppose, for instance, that the service is being delivered to students at a university. Do part-time students count the same as full-time students? What about faculty? Administrators? Parents? To avoid confusion and pricing disputes, definitions regarding who a "user" is must be handled with care.

The SaaS agreement should also make clear when the payment obligations begin: upon activation of the service, or upon usage? Most SaaS applications require payment upon activation, but ambiguity in this regard can lead to misunderstandings and soured relationships.

4. Skipping over integration issues.

If the SaaS agreement fails to address the question of who is responsible for handling any integration or configuration issues, you can bet that the customer will assume it's the vendor's job. That means the agreement needs to spell out what integration burdens the vendor is willing to take on, and at what price.

The agreement needs to spell out what integration burdens the vendor is willing to take on, and at what price.

Providing services such as business-process alignment, optimizing the use of information, application support and integration with existing systems are all important value-added services that the vendor can provide. But if these services are viewed by the vendor as premium add-ons as opposed to being bundled with the basic service subscription, the agreement had better say so. Typically, these services would be provided on a time and materials basis at the vendor's then-applicable rates, pursuant to a work order.

By the same token, the agreement should make clear whether training and support are "baked into" the cost of the service or require additional payment.

5. Failure to obligate customers to accept updates/new versions.

One of the prime advantages of the SaaS business model is that the SaaS company can easily analyze what applications are working well and which need improvement, and quickly implement software updates in a timeframe that would be impossible with on-premise software. Rolling out updates is simple, because changes are required only on a single platform at centrally located servers. Quarterly, monthly and even weekly updates are possible.

But what if a customer decides it doesn't want the upgraded software, and prefers to remain with the legacy system? It's not practical for a SaaS vendor to maintain multiple versions of its application depending upon customer whims, and so the agreement must be clear that all updates, bug fixes, upgrades, etc. must be accepted.

6. Dangerously broad and one-sided indemnities.

In an effort to reassure potential customers who are dipping a toe into SaaS waters, vendors are offering to indemnify them to the max. For example, a simple but potentially lethal indemnity clause would require the vendor to hold the customer harmless against any damages resulting from the vendor's breach of the agreement, which would expose the vendor to unlimited liability. A more sensible indemnity would be limited to claims of intellectual property infringement, with the usual carve-outs for claims arising because of modifications made by the customer, use of the service other than as specified in the agreement, etc.

Overall liability should be capped at some sensible level – e.g., the fees paid by the customer over the past year. Any exceptions to the liability cap should be reviewed very carefully. If indemnity obligations are excepted, and defined broadly, they will effectively eclipse the liability cap, because virtually every possible liability would be deemed an exception to the cap.

The customer should be required to step up and indemnify the vendor under some circumstances. The most common cases are where (1) there is a claim that the customer's data infringes the intellectual property rights or other rights of a third party, or (2) the vendor is subjected to legal proceedings for the purpose of obtaining customer data. Thus the SaaS agreement should explicitly state that the customer is responsible for the quality and legality of its data, and will cover any vendor costs if such data creates problems.

Vendors should not be required to give up their "crown jewels" (software source code) lightly.

7. Ignoring worst-case scenarios.

At the beginning of a vendor-customer relationship, both parties naturally are focused on the positives, and are loath to imagine doom-and-gloom possibilities. But an important role of the written agreement is to deal in "what if" scenarios. Such as: what if the vendor goes belly up? What if the customer simply refuses to pay?

In such cases, the innocent party will want the ability to terminate the relationship and seek recovery of damages. If the vendor appears headed for bankruptcy, the customer is going to be concerned not only about recovery of data, but whether any third party might be able to take over from the failing vendor (a so-called "rollover" arrangement). A vendor that is willing to address these issues up front in the agreement will help reassure its customers that they won't be left stranded if disaster strikes.

That also raises the question of escrow. If the vendor agrees to put its software into escrow for the benefit of its customers, it should take pains to ensure that only extreme events can trigger release of the software. Bankruptcy would qualify; simply failing to meet service levels, or even a material breach of the agreement, would not. Vendors should not be required to give up their "crown jewels" (software source code) lightly.

8. Forgetting about tax issues.

Software sales are generally subject to state sales tax. But what about software made available as a service?

The answer is unclear. If viewed as a "lease" of software, it is likely to be taxable in the state where the software is located, regardless of the location of the end user. Even if viewed as pure "service," some states tax telecommunications services, information services and even the broad category of business services, and so will be tempted to throw SaaS into one of the taxable categories.

It's not reassuring that states are particularly hungry for revenue right now – witness New York State's recent decision to impose sales tax on Amazon.com, which Amazon is challenging as unconstitutional. To be safe, the SaaS agreement should clearly put the burden of paying sales and other taxes on the customer.

9. Overusing "legalese."

Of course, we all love legal language. Why merely "give" something to somebody when it can be "granted, bargained, sold, conveyed, aliened, enfeoffed, released, confirmed, assigned, transferred and set over"?

But even in the legal realm, plain English does have its place, and clear expression is especially important when describing a service like SaaS that is still novel to many potential customers. Customers that may already be feeling some trepidation as they enter the SaaS world are hardly going to be reassured by impenetrable legal prose.

The SaaS agreement should be drafted with business people, not lawyers and judges, in mind. The agreement should reveal at a glance (1) what the service is, (2) how it will be provided, (3) how it will be paid for, (4) what standards the SaaS provider will meet, (5) what the respective obligations of the parties are (including any service level commitments), and (6) what happens when a party fails to meet its obligations.

No doubt the SaaS industry will go through growing pains as it claims a greater and greater share of the overall software market. Clear and simple legal contracts that cover all the bases are an important tool for easing those pains.

About the Author

Michael Whitener is a principal and co-founder of VistaLaw International, based in its Washington, D.C. office. Michael's legal practice focuses primarily on international corporate transactions, assisting software and other high-technology clients develop their global businesses and protect their intellectual property at home and abroad. Michael is the author of *Creating Software Alliances* (2007) and co-author of *Corporate Governance for New Directors: the Basics and Beyond* (2008), both published by Aspatore Books. Michael can be reached at **mwhitener@vistalaw.com**.

About VistaLaw International

VistaLaw International, with offices in Paris, London and Washington, D.C., is a legal services firm created by former in-house counsel dedicated to providing practical, cost-effective legal advice to global companies. VistaLaw's services include corporate governance and structuring, drafting and negotiation of commercial contracts, and handling of merger and acquisition transactions. For more information, visit **www.vistalaw.com**.

VistaLaw International LLC
International Square
1875 I Street N.W. Fifth Floor
Washington, DC 20006, USA
Tel: +1 202-429-5526