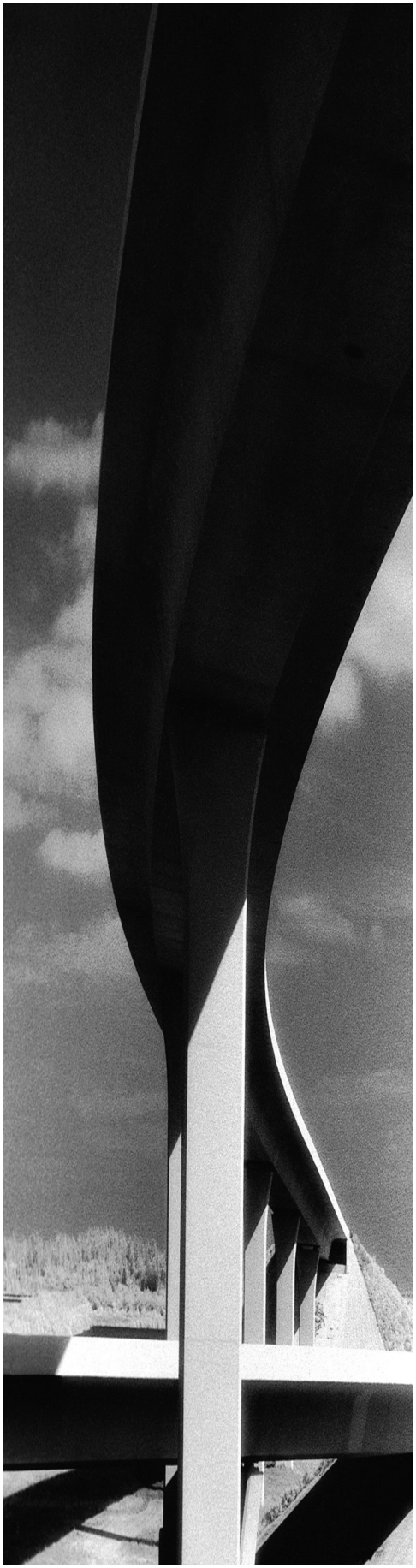

CHAPTER TEN

ASP Customer Service and Technical Support

This chapter discusses

- Service provision
- Implementation estimates
- System sizing
- Platform preparation
- Client preparation
- Customization
- User policies
- Data conversion
- Testing and quality assurance (QA)
- Training
- Going live
- Maintenance and upgrades
- Monitoring and reporting
- Call centers
- Billing and mediation
- ASP CRM

Several factors contribute to the unique nature of ASP customer support. To begin with, in the ASP business model, the ASP guarantees near-bulletproof operation of the hosted application. So you expect that very little will go wrong and, if it does, that it will be fixed without untenable interruption of your core business. Obviously an ASP must support more than an application. It's also responsible for servers, the

enabling software platform, storage, network, and physical infrastructure such as the data center.

The scope of support is much greater than for a conventional IT company like a software vendor, whose help desk personnel need address only a fairly predictable set of problems—things like software patches, hardware compatibility, upgrades, and so forth. With installed applications, especially mission-critical ones, the customer usually keeps support staff on-site or on-call around-the-clock to enable prompt troubleshooting. For the most part, ASP support personnel are not located on-site at the customer premises. In an ASP virtual support environment, staff are much more dependent on monitoring and remote support equipment to solve problems quickly. Obviously, *that* equipment, as well as the hosted application and other components of the total ASP solution, must also be absolutely dependable. Of course, because an ASP is the sum operation of multiple software, network, and other players, the partners are dependent on each other for reliability. And although partners typically have equity relationships or SLAs that provide them incentive to promote top performance, nevertheless, one broken link in the ASP chain can undermine all others. Players like telcos partnering with software vendors, for instance, might not be expert at supporting complex software, even though they are extremely qualified at troubleshooting problems with voice networks.

To complicate matters, the nature of IT support in general has become multidimensional. Call centers, for instance, increasingly handle queries by phone, e-mail, fax, and whatever other medium the customer prefers. They prioritize problems and address them accordingly. They use workflow and other systems to route specialized queries from the customer service representative (CSR) to, say, a database expert. They are increasingly aggressive in marketing their products and services to customers at every opportunity. For instance, if a CSR knows a new software release is ready, he may refer a customer to the sales department to buy an upgrade for $50 instead of to a database administrator for temporary troubleshooting. Yet customers expect all queries to be handled promptly and efficiently. And, with comparable IT functionality becoming more generally available from multiple vendors, vendors now realize that customer service is a key competitive differentiator.

Compared with traditional customer service and technical support, ASP customer support is *complex.* But to you, the customer, it should *never* seem so. Most of you want to call one number and have problems fixed quickly. When you call, you should be able to explain the problem very easily and have the support

person understand and deal with it. For instance, you don't want to be on the phone for 30 minutes trying to explain a database problem to a network specialist because the CSR misdiagnosed the problem and transferred you to the wrong expert.

However, with more than 1,500 ASPs operating worldwide, not all of them understand or offer full-featured virtual customer service and technical support. This is especially true of small ASPs specializing in one or two very focused applications. Often these companies were service bureaus or VARs with modest sales and support programs, and they may tend to trivialize this aspect of an IT operation. This is not to say these smaller ASPs are necessarily slipshod when it comes to support. Indeed, their narrow focus may require fewer value-added services like complex support.

On the other hand, customer service—like security, storage, and infrastructure back ends—will likely become more outsourced and specialized as the industry matures. One call center may support six different ASPs by using the ASP's staff or by training their own. Such personnel could easily handle these low-level problems and pass on more difficult ones to an in-house support department at each ASP. Infrastructure service providers and MSPs will also absorb many platform monitoring and maintenance tasks from ASPs. It's not unlikely that they will specialize in supporting certain ASP operating systems, hardware platforms, networks, applications, and, possibly, vertical markets. However, although diversification and specialization are inevitable in any industry, the ASP industry will have to counterbalance that tendency with its marketing promise of providing a simple, all-in-one solution on a single bill.

Up-front Customer Service

ASP customer service starts as soon as you log on to the ASP's Web page or call its sales department to inquire about leasing an application. If you are inquiring about pricing and deployment time and it's not clearly conveyed by the Web site or by the sales rep, then that shortcoming may be indicative of the service your hosted application will receive later.

If you decide to lease, you can subscribe to out-of-the-box applications, like hosted e-mail, usually right on the Web site by virtually signing up and choosing your price/service level like Silver, Gold, or Platinum. Service can be up the same day. More complicated enterprise applications that require integration may take months. Many turnkey enterprise applications, though, require much less

preparation, and subscribing is relatively quick and painless. Part of the subscription process also involves defining the terms of the SLA. Many ASPs offer standard SLAs that you can't amend, whereas others will negotiate special terms.

Then the ASP must provision the service by setting up, turning on, or ordering the following components before deploying the application:

- Network—Capacity, quality of service, connection to customer via local provider, and so on
- System platform—Extra storage devices, client installation/upgrade, middleware, and so on
- Application—User names, and so on
- Servers—Shared or dedicated, disk capacity, chip power, and so on
- Billing—Customer billing data, and so on
- Reporting/monitoring—Real-time or historical, the metrics to be monitored, and so on
- Security—User profiles and policies, and so on

This activity, however, is still preliminary to the service relationship between you and the ASP throughout implementation.

Hosted Application Implementation

The ASP implements the hosted application in roughly chronological phases. First, the ASP has to decide the functionality that is needed to meet your business requirements. If you lease a turnkey, generic hosted application, the ASP can commit to a predetermined schedule. After all, little customization between legacy and hosted applications is needed. However, if customization is required, the ASP must estimate the extent of it and work out the fee it will charge you. For example, the fee might be based on time and materials for the whole project or be a flat fee for specific work.

Next, the ASP must size your solution. This phase, too, can be less or more complicated according to different factors. If the ASP makes you conform to its own system platform, sizing is easy. You have already decided to forego customization and will adapt your work processes to the hosted package. If you require that the generic application be customized, then the ASP must tailor the software to your unique specifications.

To size hardware, the ASP must decide the number, type, and use of computers to be deployed, and the system architecture of the solution. For instance, will different types of servers be used as firewalls, database, and Web servers? At this

phase, the ASP also will typically decide on disk space for each computer and storage capacity for storage devices. This estimate should take into account user data accrued over the period of the contract. The same goes for servers. The ASP must estimate "headroom"—how many additional servers and how much rack space you'll require—as well as whether the servers will be dedicated to you or shared by you and other customers. Any hardware architecture decisions like whether to cluster servers should be made at this point. And, of course, the ASP must choose processors powerful enough to drive those servers.

Actual platform preparation usually involves loading the operating systems, partitioning drives (logically dedicating drive capacity in the same server to different users, departments, and so on), loading the servers and storage devices into the racks, and deploying network firewalls at your customer premises and the data centers.

Client Preparation

If the leased application uses a client/server architecture, then the ASP must load the client software on your on-site desktops. If it uses thin clients, then the ASP must load the appropriate communication protocol on them.

If you've contracted for VPN services and are deploying encryption software on your own users' PCs (not in a router or other device), the ASP loads the appropriate security software on the clients now.

Finally, if you've elected to have the ASP remotely manage the application so it can do things like virtually add, delete, and change users, the ASP must now load that software on the clients.

Customization

Although ASPs roughly configure the application to your requirements when they install it in the data center, if you require customization, the ASP makes the actual code changes at this stage.

As mentioned earlier, customization ranges widely in the extent of its complexity. For instance, you might want your legacy applications integrated with the hosted application. Obviously, this cannot be done from the data center, and it requires expertise in additional applications. Also, you might not want your mission-critical applications inextricably tied to a hosted application that you may discontinue using in the future. It's up to the ASP to gauge the type of APIs

and tightness of integration to your level of commitment to the solution. Depending on whether the ASP offers integration as a value-added service, and often even if it does, these charges are additional to the leasing fee. Some ASPs will bundle integration into the fee, but this is likely only if the required time and materials are easy to determine.

Typically during this phase, you experience for the first time the ASP's value-added services like outsourced IT personnel and accelerated application deployment. So customization is a key indicator of the quality of the ASP's future customer service.

SUCCESS STORY

IBM Global Services
Customer: CareTouch, Inc.
Key elements of the solution: Accelerated deployment, extensive application development and integration, one-stop shopping for the complete solution

An estimated 52 million people in the United States today have taken on the responsibilities of life care: that is, caring for sick loved ones with a continuing health condition who cannot live fully independent lives. Often these caregivers have no prior experience. Yet, the current healthcare system, its resources strained to the limit, provides little support in this vital area. This leaves an even bigger burden directly on individuals and families and indirectly on their employers. For example, lost productivity from employees caring for elderly relatives and friends cost American business between $11 and $29 billion yearly.

CareTouch, Inc. offers a powerful remedy to reach and support this often-isolated group of caregivers. The company was conceived in January 2000 by the world's largest not-for-profit healthcare delivery organization, Kaiser Permanente. Moving at Internet speed, CareTouch launched its call centers and the most robust, comprehensive e-commerce site of its kind www.carepanion.com six months later. In fact, it built its Web

continued

e-commerce solution in just 85 days with the help of IBM and the use of leading IBM WebSphere platform technologies such as WebSphere Application Server, WebSphere Commerce Suite, and VisualAge for Java.

"We saw a huge, unmet need in the healthcare marketplace and we're using the Internet and IBM e-business technologies to meet this need—there are a number of Web companies out there trying to do a part of what we're doing, but not with the same speed or scale," explains Dr. Peter Juhn, MD, president and chief executive officer. Prior to founding CareTouch, Dr. Juhn was the founding executive director of the Care Management Institute, Kaiser Permanente's national disease management, outcomes measurement, and clinical policy entity.

A Robust, Scalable e-Business Solution

Speed-to-market is essential for any new company. Still, CareTouch had to make sure it built a robust e-commerce foundation that could handle future growth. Through its parent company, Kaiser Permanente, it would have instant access to more than 8.5 million potential customers. Down the road, it could potentially reach a worldwide audience on the Net.

To get it right, the CareTouch executive team decided to work with a technology partner to launch the Carepanion e-Commerce System. After a spirited competition among several leading firms, it chose IBM and its WebSphere platform solution. Scalability and robustness were the decisive factors. As Juhn says, "We are planning for success—we fully anticipate we'll have millions of customers using our site on a daily basis."

In fact, IBM Rochester benchmarked the system prior to its launch, showing it could readily support up to 2,500 concurrent users with its current architecture. And, it can scale up easily to support double that number.

"With the architecture IBM developed, we simply add more servers to support more users," adds Dr. Prasuna Dornadula, vice president, Web development, and acting chief technology officer. "With WebSphere Application Server and Commerce Suite," he continues, "we have a proven solution that has been used to build a number of major e-commerce

continued

sites—we consider it to be more robust and easily scalable than any other alternative we considered."

A Proven Technology Partner

Beyond the technology, CareTouch also had high confidence in IBM as a partner. IBM had already proven its mettle in helping Kaiser Permanente implement a Clinical Information System. According to Dr. Eric Aguiar, MD, executive vice president and chief business officer, "It was an 'A' team and that's what we needed to jump start our project—time is serious money in the dot-com world."

Not only that—the Kaiser brand name has been carefully protected over the years and represents a huge corporate asset. The company wanted to ensure that its e-commerce solution would continue to maintain the value of that brand. CareTouch was counting on IBM's expertise to ensure they got it right from the word *go.*

"IBM has repeatedly demonstrated it has the ability to ramp up not just the development effort, but to scale a solution to support hundreds of thousands and potentially millions of users," says Aguiar.

A Comprehensive, End-to-End Solution

The time was early May 2000. The target launch date was late September. After six weeks of intense design meetings, the joint CareTouch and IBM Global Services team went to work. CareTouch developers worked with Jasper Design of New York to create the look and feel of the Web site. IBM, meanwhile, focused on building the e-commerce engine and doing back-end database integration.

CareTouch briefly considered an Oracle database as part of the end-to-end solution, but, says Juhn, "We concluded that for overall robustness and compatibility with other applications, DB2 was our best choice—we couldn't afford to get hung up down the road trying to work out an integration problem."

"Time-to-market was key," reiterates Jeff Lucas, vice president, sales and market development. "One of the reasons we chose IBM was that it gave us one-stop shopping with access to today's leading IBM Web technologies. We could use a DB2 database and the WebSphere platform

continued

environment, including VisualAge for Java, and it's all designed and pretested to work together smoothly," he adds. "In addition, IBM could provide the hardware, Web hosting services, and the integration services needed to pull it all together. Frankly, without the IBM total solution, we wouldn't have been able to hit our September 30 launch date."

A Highly Productive Development Environment

The Carepanion e-Commerce System runs on heavy-duty IBM RS/6000 servers located at IBM's server farm in Rochester, New York. Approximately 75% of the application code was based on WebSphere Commerce Suite. IBM Global Services developed the remaining code using VisualAge for Java Enterprise Edition, deployed on WebSphere Application Server.

"WebSphere Commerce Suite gave us the built-in functionality we needed to drive credit card transactions through our e-commerce engine," says Dornadula. "We estimate it saved us anywhere from two to four weeks' worth of development time and reduced the cost of building the solution by approximately $1.4 million," he continues. "All we needed to do was tweak the interface to link it up with the CyberCash system, a leading payment solution," he adds. And, he concludes, "We got additional support for this work from the IBM Toronto Lab team which helped us move forward quickly."

CareTouch used IBM's formal project implementation approach to manage the project. It's an iterative approach with overlapping phases that gives developers the opportunity to respond to changing conditions and scenarios. The phases include:

1. Requirements definition/macro design
2. Micro design
3. Build and test
4. Production readiness and deployment.

A team of four IBM and four CareTouch developers used VisualAge for Java to develop additional functionality, including a Java-based calendar function, care registry, and address book. Caregivers can use these resources to schedule home care services for their loved ones, share

continued

information with family and friends, and also reach out to a wider community of people facing similar challenges.

"We looked at alternative Java developer toolkits," says Dornadula, "but we found VisualAge for Java offered superior debugging capabilities. Sometimes building interfaces can be a challenge," he adds, "but with the integration of VisualAge for Java with WebSphere, it was very easy—we estimate it reduced our development effort by an additional 10%."

In building future applications, Dornadula anticipates his team will be able to reuse at least 50% of the code developed for the Carepanion e-Commerce System. In addition, they are planning to use Enterprise Java Beans to further improve reuse and achieve significant cost savings. For their next project, the CareTouch team is evaluating the feasibility of developing a wireless capability, which will allow caregivers to access Carepanion via palmtops and other portable computing devices.

A High-Revenue e-Business

In its first year of operation, CareTouch anticipates it will generate millions of dollars of revenue through targeting three distinct areas of need. Juhn calls these "our corporate ABCs" and they include:

1. Advocacy support, giving patients and caregivers a forum to get the advice and assistance they need to make important healthcare decisions—for example, the site provides tools members can use to build a CareCommunity that brings together patients with similar problems.
2. A brokerage function, which leverages the buying power of large numbers of consumers to help people get healthcare products at special discount prices in a business-to-business environment.
3. Concierge services, which help caregivers put together all the essential support services they need, including grocery shopping, home cleaning, in-home nursing, and more.

"Caregivers are often thrust into the role without prior experience or an instruction manual," says Juhn. "They must provide care services, seek

continued

information, investigate, and shop for and buy medical and health products that are available to solve the needs of the patient," he continues. "The Carepanion e-Commerce System is especially designed to put them in touch with all the resources they need to navigate through what can be a fragmented, confusing world of options and find the most effective, economical answers available to them today," he adds. So, he concludes, "The fact that we have partnered with IBM to build this solution will help us deliver the highest level of service to these people as they face the enormous challenges of caregiving."

Permission courtesy of IBM

User Policies

Policies determine who can use what functionality and when. There is some leeway in which administrators—the ASP's and/or yours—will create and/or enforce them.

Policies affect both networks and applications. Network policies are an integral part of traffic management. Via network policies, network administrators grant different types of traffic like data, voice, and video from different companies or departments greater or lesser availability and bandwidth as their users require. Although you have input into the network SLA, once it's determined, the ASP or its network supplier enforces network policies.

For instance, the network section linking a company's inventory warehouse with its sales force might need 24-hour availability and T3 bandwidth, so orders placed by salespeople are filled in time by the warehouse, whereas a customer service engineer troubleshooting on-site equipment problems might need only a wireless link from 8 AM to 7 PM on weekdays.

Administrators can also create policies so specific that only certain users are allowed to access certain application features over the network, and only at certain times of the day. If a large bank processes all the preceding day's checks from midnight to 4 AM the following morning, then the administrator can prioritize appropriate network sections for that purpose during that period.

Policies affecting applications authorize certain users access to certain applications or application resources like databases or screen fields. A sales rep, for

example, might be able to enter an order on a screen but not have access to the *Approved* field. Only his manager does because she has to approve all orders. Conversely, the manager may not have access to the *Credit Limit* field on the same screen because, when the sales rep enters the order, the order entry system automatically fills that field with the amount by which the customer has exceeded his credit limit.

As with network policies, you have input into application policies to determine the SLA, but afterward the ASP enforces those policies. However, if you specify, some ASPs will permit your IT staff some administration authority like adding new users. Obviously, however, the ASP must be promptly alerted when such changes are made.

Data Conversion

SMBs without in-house IT departments may fail to take into consideration the often necessary step of data conversion. Larger companies are probably experienced in the process because they've run different applications in different departments and had to load data between applications and reformat it during the process. In the ASP model, data must be loaded from your legacy-installed applications to the ASP's hosted applications.

If it's transferred between similar systems from the same vendor—say, from an installed SAP ERP system to a hosted SAP ERP application containing identical modules—then conversion isn't needed. The data is typically copied to some storage medium at your premises according to the ASP's specifications, and loaded onto the hosted application in the data center. If it must be converted from one or several formats into a format usable by the hosted application, then actual data conversion—rather than data transfer—takes place.

ASPs either outsource conversion or do it themselves. If they outsource, then they hire consultants, service bureaus, integrators, or other service providers to handle the process, and then the ASP loads the converted data into the hosted application. The ASP might pass on the fee for the service to you as a charge independent of the leasing fee. For instance, a pure-play ASP that offers no value-added services would likely outsource conversion. If the ASP does the conversion, it assigns its own staff to go on-site and either actually do the conversion or oversee it to make sure you convert the data according to the ASP's specifications. An FSP offering integration as a value-added service would likely be efficient at conversion, do it profitably, and be able to bundle the cost into the overall leasing fee to your satisfaction.

Testing and Quality Assurance

ASPs must maintain a separate testing environment to "test drive" hosted applications loaded with your data on a computing platform like the one it has sized and configured for you. The test environment is like a Y2K platform where, after an upgrade, applications are test run to see if they are Y2K compliant. Most test environments are fairly standardized, not customized literally to replicate the hardware and other equipment of each new customer. For instance, if you lease an application for 200 users, the ASP can create a simulation environment with 200 "logical" seats, not actual PCs.

However, the ASP does use the testing platform to configure and customize the actual hosted application, as explained earlier, and then to run it through various user scenarios to see if it can hold up to the performance guarantees of each customer's SLA. The test driving is generally what's considered "testing."

Training

Training your users is an ongoing process, but initial training should take place before the hosted application is used for actual business. To get good training, you may stipulate that course materials and presentations cover any functionality, tasks, and policies uniquely created by the customized hosted application. This means that standard materials from the software vendor of the application should be revised accordingly.

Generally, too, the best training offers content in several modes—written, presented, and in interactive workshops—so that users who learn better in certain modes are exposed to the content in their preferred mode. Interactive workshops also expose users to actual common screen sequences that occur during routine work scenarios. For instance, sales reps who will have to create orders using a sales automation package usually benefit by doing so in a demonstration rather than having to transfer knowledge they learned from screen shots in a workbook. This said, multimode training is nice to have; not a "must have."

Train-the-trainer strategies are also a cost-effective way of shortening the ASP's period of involvement with your users and ensuring that ongoing training occurs after the initial sessions. Using this approach, ASP personnel train one or two users who then train the rest of your users. It's a good idea to choose initial trainee/trainers who are proven managers or who excel in the area of expertise that the hosted application addresses. Often their feedback helps the ASP make the application more user friendly and to more accurately handle the company's

work processes. Also, their proven competence makes them the best candidates for reinforcing "the right way" of using the application to your other users in the future.

Often, to expedite an installed application's launch, training occurs while the application is still being tweaked during the final stages of customization. This may be necessary to cut costs and stay on schedule, but it can be very frustrating for users who may encounter different functionality in the finished application than in the version on which they were trained. With hosted applications—especially in the case of accelerated deployment—this may be necessary. But the consequences for your users will probably also be more harmful. A bulletproof application is the best training ground, and ASPs should always strive to approach that ideal.

Although during customization your "techies" develop a relationship with the ASP (this is where an ASP first proves its "tech" competence), during training your users develop a similar relationship (this is where the ASP proves it has a nice personal "touch"). Although the ASP model stresses a high-tech/low-touch approach, if an ASP takes a "no-touch" approach, it can backfire. For instance, the ASP might ignore your users' input during training, so even if the live hosted application works fine later, the users' resentment from training may undermine their acceptance of the application. This hurts their productivity and your bottom line. The importance of this cultural factor to the long-term success of your relationship with the ASP cannot be overstated.

Going Live

Going live is a matter of the ASP replicating the customized application from the testing environment to the production environment in the data center and, if needed, at your site. Once the hosted application is live, you should be able to use it to conduct legitimate business.

Routine Maintenance and Upgrades

As a fundamental tenet of the ASP business proposition, most maintenance and upgrades are included with the leased application. How much of the cost for these activities is included in the leasing fee and how much is a billable service depends on the ASP. Some ASPs may charge a premium for especially redundant

backup or for extra storage, for example. As a rule of thumb, though, these activities come part-and-parcel with the hosted service.

You should also expect certain key upgrade activities to be covered by the maintenance/upgrade agreement, have input on the date of upgrades, and expect the ASP to take sufficient change management measures to ensure the upgrades do not unsatisfactorily interrupt work. The ASP should routinely upgrade, either remotely or on-site, needed client software as well as the operating systems on the servers in the data center. It should also locate and install or develop patches for all software bugs in its vendor's software. (If the ASP has an equity relationship with the vendor, the vendor is likely to be more responsive in creating patches.) The ASP should also keep at your reasonable disposal a database administrator certified on the databases used by the hosted application.

Of course, the ASP should also routinely backup, restore, and archive data. The media on which data is stored is determined in the SLA, but most ASPs will back up data each night and store an extra copy on tape off-site. Some may offer less frequent backup intervals for lower fees.

Technical Support

Users take it for granted when an application runs smoothly, but it's usually the harried activities of the IT department that ensure bulletproof performance. By hiring an ASP, an IT department is often seeking a way of relieving the pressure on its own personnel to support users. In many cases, unlike users who might gripe incessantly about poor technical support (but have little power in improving the situation), IT managers might scream once or twice and then just pull the plug on the hosted application. If your ASP fails to address technical support, it's a serious problem.

Most of the major areas of technical support like upgrades, backup schedules, and SLA monitoring have been discussed to some extent in other chapters, but certain procedures deserve elaboration here.

Monitoring and Reporting

Monitoring and reporting, whether done by the ASP itself or outsourced to an MSP, is, ideally, proactive. The ASP should be so adept at scrutinizing the performance metrics that it can predict, say, network congestion, warn you, and fix it before service is affected. Such foresight comes with increasing experience with each customer, but the ASP should also have developed a "feel" for the

application during the earlier "testing" phase. Similarly, the ASP should be able to run test scenarios (say, in off hours or on the testing platform) to determine whether system performance will degrade in the event you need to upgrade the application, for example, to accommodate additional users.

Any ASP should monitor most key components of its infrastructure like network, servers, databases, and applications. However, the ASP should also never overlook any equipment integral to the application's performance—like a SAN in the data center. Routine maintenance will usually also entail activities like managing storage and other assets, as well as administering user access.

Reports submitted to you corroborating SLA performance guarantees must also contain data that's *relevant to your typical production work periods.* For instance, an ASP submitting a measurement of transactions per minute taken at midnight when four users were on the application is of no use to a company performing production work from 9 AM to 5 PM, when the system has to accommodate 500 users without performance degradation. Test results from artificially generated data scenarios are also not accurate enough to be relevant.

Of course, monitoring should also not interrupt the users' work or the application's performance.

The ASP should generate reports analyzing *all* relevant performance trends. These may vary from customer to customer. An FSP may have to monitor transactions per second for a customer leasing an e-commerce application but may not have to for a customer leasing a messaging application. Monitoring and reporting systems should scale as the application scales and be readily integrated with the appropriate help desk or other departments whose staff will troubleshoot problems.

Call Centers

As mentioned earlier, you should be able to contact the ASP support call center in multiple ways like phone, fax, and e-mail. Most ASPs will offer 24/7 support, sometimes for a premium fee. This said, though, the predominant mode of contact is phone calls to live agents.

Ideally, an ASP call center should enable the agent to access your service history, including all services you are leasing, all bills and payments you've made since services were turned on, and all previous support interactions (from users' questions about billing to actual troubleshooting). Some ASPs provide agents copies of questions that customers commonly ask, and suitable answers and

actions for solving common problems. Some call centers also keep logs of how problems were escalated from agents to experts and record the actions taken at each stage of escalation. Such preparations expedite near-term service and refine call center business processes over time.

If the ASP uses multiple call centers or subcontracts some of its support to an outside company, you should be able to call one toll-free number and quickly reach the appropriate personnel at all centers.

Some ASPs will also place FAQs and responses and other information on their Web sites so your users can conduct their own self-service for low-level inquiries. This frees call center agents to deal with more substantive customer support issues. If the ASP lets you add, delete, and change users and otherwise mutually administer the application, then it should also offer some method by which you can do so remotely from your premises.

Billing and Mediation

Two aspects of billing are unique to the ASP model—consolidated, itemized services from multiple providers on one bill and tracking application activity to calculate usage-based fees. Contact center agents with access to your billing/payment history can probably handle general billing inquiries, but for rebates and other areas of negotiation, your ASP should probably assign a representative—either an agent or some other form of account manager—who acts as the primary liaison between you and the ASP. Most customers appreciate it when an ASP "puts a human face" on their virtual relationship. You will probably find that a single rep reinforces the sense of accountability and single-source service that define the ASP model.

Because the model is virtual, you may expect to be billed in the same mode that you lease services—over the network—although all ASPs may not offer this option. If they do, they may institute common e-business billing practices like electronic bill presentment and payment, and electronic funds transfer. If the ASP outsources bill presentment, the final bill should only represent the ASP, not the subcontractor. If the ASP is leasing different applications to different departments and you prefer that each department pay for service out of its own budget, then the ASP should be prepared to bill each department accordingly with separate statements. Of course, these practices hold true for ASPs offering traditional billing methods as well.

Usage-based pricing models require that the ASP track your activity for the relevant equipment such as database, storage, and scanners used to deliver all

services. This pertains not only to per-click pricing, but it is necessary also for pricing models like "variable/tiered" ones where you're allowed a certain amount of usage within the price tier and pay a premium if you exceed it. Key service components like average network bandwidth, transactions per second, and response times are captured as a matter of course in the application monitoring process. However, particular usage metrics for billing purposes require actual, not average, data records and monitoring equipment tied to specific pieces of equipment like, for instance, scanners (for recording number of documents scanned), databases (for number of transactions), clients and servers (for number and initiator of retrievals), and so on.

Tracking usage is complicated. The ASP may have to track multiple metrics for multiple pieces of equipment like network, clients, different types of servers, storage devices, and so on. It may also have to identify the user or department that initiated an activity like a document retrieval. When the hosted application is bundled with other hosted applications or integrated with your legacy applications, the ASP may track the interrelated activities of multiple applications where certain actions initiate other actions for which you're billed—for instance, a business transaction in an e-business application might involve a document retrieval in a workflow application.

General Expectations

Although certain ASP players like software or hardware vendors may not traditionally excel at service, if they are involved in an ASP coalition, their partners should. Inasmuch as ASPs offer services virtually, and CRM has become a common practice, you may differentiate ASPs according to how well they carry out major CRM practices. Although not mandatory, ASPs may aspire to achieve CRM benchmarks to be more competitive.

CRM is the process of combining an ASP's best business practices, optimized work processes, apt technology, and relevant knowledge to service customers better and retain their business. However, CRM is not merely a means of guarding the customer assets that an ASP already has. The business intelligence and opportunities for customer interaction that it engenders also help the ASP more effectively market to customers and grow the services they use. This attitude transforms customer service from a reactive chore into proactive prospecting for increased business. Although this is good for the ASP (it transforms customer service from being a cost drain into being a profit engine), it is good for you too

(it uses the ASP's basic business motive, profit, to promote superior customer service). Accordingly, although an ASP may win more of your business, your value for your dollar usually increases dramatically via bundling discounts and other bonuses.

To begin with, ASPs should minimize confusion by designating which of their employees may have contact with you. If certain IT personnel worked for months with your IT staff to customize the application, the ASP might capitalize on that personal relationship by also having one or more of that group do ongoing troubleshooting for you. Similarly, specific call center agents, customer account managers, and other liaisons often deal consistently with the same customers. These more consistent and intimate human encounters, whether it's an account manager solving a problem or a sales rep making a new service suggestion, are known as "moments of truth" in the customer interaction cycle. By exploiting them, ASPs keep you happy in the short term and keep your business in the long run.

By managing the customer interaction cycle from end to end with appropriate technology, the ASP can humanly "touch" you more often and more efficiently *and* boost its bottom line. For instance, an ASP might use integrated telephony tools to enable customer service personnel not only to communicate with you but also to access your service and billing history and to compare notes to satisfy you better.

Minimizing wasted time is also key to superior customer relations. Studies have shown, for instance, that the average user can tolerate about a two-second response time in applications they use regularly. Greater delay usually upsets their work rhythms so that, for instance, a manager approving loans forwarded to her in a workflow system might forget why she accessed a document, especially if phone calls interrupt, by the time it appears on the screen. The cumulative affect of such delays undermines service personnel efficiency.

Likewise, studies indicate that customers contacting a call center can tolerate about a four-second wait to speak to a representative. Most professional call centers make it mandatory that agents pick up before the third ring, for example. However, if they then put you on hold for 10 minutes, they've ruined a good first impression. Instead of putting you on hold, agents should always direct you to an expert who can resolve your issue. Prompt communication is especially important in an ASP setting because your user cannot physically contact the ASP. With installed applications, IT personnel and users tend to get to know one another and users will, if needed, walk down to the IT department and get someone to fix

a time-sensitive problem. Users of hosted applications have no such outlet for their frustration.

The more methods an ASP uses to *quantify* its staff's customer service activities, the better. The ASP can offer bonuses to individuals who excel in this area, assign them to the busiest shifts in the call center, use them to train less adept agents, and so on. For instance, if—to resolve issues—an agent averages three minutes per customer call and rarely hands off to other experts or rarely makes multiple callbacks to customers, you have no grounds for complaint. However, if another agent averages 15 minutes per call and makes excessive handoffs and callbacks, then they are wasting your time.

Although putting the best agents on the busiest shifts is generally a good call center policy, putting more of them on those shifts is usually critical. For your monthly fee, you expect an ASP to have done its homework and to have calculated the cost of expedient customer service into the price of the service. By doing things like maintaining records of your behavior over time, ASPs can better predict busy periods to plan adequately for them.

Putting a face on virtual service promotes intimacy, and intimacy usually leads to customer trust and a long-standing business relationship. ASPs should pursue relationships as well as sales. Not only is this a fundamental premise of customer service, but CRM studies have shown that a long relationship with one customer is much more profitable than separate brief relationships with multiple customers. The marketing, administrative, and other costs associated with winning new customers or winning *back* dissatisfied ones can hurt both the ASP's bottom line and—as important for ASPs balancing their own payments to their service providers—its cash flow. The same goes for you. Changing ASPs, although easier than overhauling installed systems, drives up your costs and wastes time.

A related point is also true. In most cases, ASPs will give superior service to the 20% of customers that bring in 80% of their revenues. This is not to say that other customers get *poor* service as a result. Often, to avoid this, ASPs will offer tiered customer service plans—like Silver, Gold, or Platinum—so that customers can purchase different levels of customer service according to their budgets. For instance, Platinum might get a customer a phone response from an IT expert within five minutes of the inquiry and a guarantee that any application problem will be fixed within one hour of the call. Gold and Silver levels might merit a less-prompt response. The ASP may even provide incentive for customers to lease more services by offering Platinum customer service at the cost of Silver if they spend a minimum amount monthly. Neither strategy shortchanges you unless the resulting service proves insufficient to meet your anticipated requirements.

You have to remember, ASPs are in business to make money. If your excessive service demands consistently outweigh the revenues you bring the ASP, it's likely the ASP will pay less attention to them in the hopes you'll sign on with someone else.

You may save yourself a lot of trouble by, in addition to checking the ASP's reference accounts, finding out the number of customers an ASP has had and the length of the relationships. Numerous short engagements usually indicate either customer dissatisfaction or frequent revisions to an ASP's business plan and strategic mission. Neither suggests stability. Relationships, on the other hand, indicate customer satisfaction and intelligent business strategy as well as commitment to a market and financial stability.

Ideally, the ASP should offer such bulletproof hosted applications that you never tax the resources of its customer service personnel. However, it's probably reasonable to assume that the more customers an ASP has, the more likely something will go wrong.

Nevertheless, you should expect customer service commensurate the requirements of your hosted solutions for the duration of your contract. If your ASP fails to understand that customer service is both your *right* and an *ongoing process,* then it does not have your best interests at heart and should not have your business long term.

Key Concepts

- Service provision—The ASP sets up, turns on, or orders network, system platform, application, servers, billing, reporting/monitoring, and security
- Implementation estimate—The ASP estimates the extent of customization and fee
- System sizing—The ASP decides the number, type, and use of computers to be deployed as well as the number of software seats; and the general system architecture
- Platform preparation—The ASP loads operating systems, partitions drives, loads servers and storage devices into data center racks, and deploys firewalls
- Client preparation—The ASP loads communication protocols and client, encryption, and remote management software on customer desktops
- Customization—The ASP makes actual code changes to the hosted software that may involve integration with legacy applications and other value-added integration
- User policies—Network policies grant different types of traffic from different companies or departments greater or lesser availability and bandwidth;

application policies authorize certain users access to certain applications or application resources

- Data conversion—The ASP or a partner converts the formats of customer legacy applications to formats usable by the hosted applications
- Testing and QA—To guarantee SLA performance levels, the ASP maintains a separate testing environment to "test drive" hosted applications loaded with customer data
- Training—The ASP should cover any functionality created by the customized hosted application and often uses train-the-trainer strategies
- Going live—The ASP replicates the customized application from the testing environment to the production environment so the customer can conduct legitimate business
- Maintenance and upgrades—The ASP should include client software, operating systems, patches for software bugs, data backup, restore and archival, and an on-call database administrator
- Monitoring and reporting—Monitoring should be proactive and should cover key components of and equipment integral to the hosted infrastructure; reporting should contain data relevant to the customer's typical production work periods
- Call centers—Customers should be able to contact the ASP call center readily (often in multiple ways like phone, fax, e-mail), and agents should have access to the customer's service history
- Billing and mediation—The ASP should present consolidated, itemized services from multiple providers on one bill; ASP mediation should track *actual* application activity to calculate usage-based fees
- ASP CRM—An ASP's CRM should combine best business practices, optimized work processes, apt technology, and relevant knowledge to service customers better, retain their business, and win new customers