

Managing Trading Partners

By John K. Ousterhout

By now, the high expectations for business-to-business (B2B) applications have become familiar. Analysts have projected trillions of dollars of B2B transactions within a few years. Companies have begun to deploy B2B integration servers, which connect to existing back-end applications, and send and receive Extensible Markup Language (XML) documents over the Internet to automate business



relationships. In doing so, companies hope to reduce costs, create new revenue opportunities, and improve customer retention. A raft of start-ups has sprouted to exploit B2B opportunities. Some are implementing new Internet-based business models such as trading exchanges and brokers; others provide the infrastructure for B2B applications, including integration servers and professional services.

But in all the excitement over B2B, a key issue in implementing B2B applications has been largely overlooked. That issue is trading partner management. Costs associated with bringing new trading partners online and managing ongoing relationships will dominate the total cost of implementing B2B. Inefficient trading partner management will be one of the most common reasons for trading community failures over the next few years. Anyone thinking about building B2B applications should be aware of the issues in trading partner management and the techniques and tools available to reduce the cost of managing trading partners.

The Integration Process

Implementing a B2B application such as supply-chain automation or electronic bill presentment typically proceeds in three phases: business process modeling, back-end integration, and partner integration. In the first phase, the Internet-based business process is defined. This requires you to reach agreement with your partners on a set of XML document types to carry information between you and your partners, along with protocols that specify how the documents are transmitted, relationships between documents, and so on. Several XML-based business processes have already been defined by companies and standards organizations, such as RosettaNet for supply-chain automation and IFX for electronic bill presentment. So, often a company can simply select an existing standard rather than define a new business process.

The second stage in implementing B2B is back-end integration. This is typically done by deploying a B2B integration server. The server contains adapters for communicating with existing enterprise applications, such as databases and Enterprise Resource Planning (ERP) systems. It also contains facilities for sending and receiving XML documents over the Internet. The B2B integration server must be programmed with business rules that connect a company's particular enterprise applications with the XML-based business process. For example, the arrival of a price-quote-request XML document might trigger business rules that invoke an ERP application to compute pricing information and then return a price-quote-response XML document to the sender.

The third stage in implementing a B2B

application is connecting with particular trading partners. Let's assume the basic parameters of the relationship have already been worked out and all that is needed is to implement them. In that case, this stage involves various tasks, such as:

Configuration — Information must be entered into the B2B integration server about how to communicate with each partner.

Security — Information for authentication must be exchanged with each partner, such as passwords or X.509 certificates; the server must be configured with access control information, specifying the transactions the partner can invoke.

Customization — Special facilities may need to be implemented for particular partners. For example, a partner might have special shipping requirements or billing terms. A partner might use slightly different names in documents, which

Select an existing standard rather than define a new business process.

will require translation. Partners may even have customized catalogs.

Testing — Once partner-specific configuration and customization are complete, the connection must be tested to ensure that it's operating correctly. Testing is difficult because it involves different organizations; neither side can easily observe what's happening in the other organization.

Once these three stages have been completed, business transactions can automatically begin to flow between the company and its partners. However, there will be ongoing costs to manage

partners. For example, the terms of a business relationship or the underlying XML protocols might change, or a partner may switch to a different integration server, or a company may decide to offer new services. All of these changes will result in additional overhead.

Trading Partner Management Costs

Trading partner management involves the third implementation stage (partner integration) and ongoing overhead. These are likely to be the most expensive issues over the lifetime of a B2B implementation. The first step, business process modeling, only occurs once per process; most companies will avoid this cost by using standard protocols. The second step, back-end integration, occurs once in each company for each protocol supported. This code can be reused for different partners. The third step, partner integration, must be repeated for each partner and each protocol. Ongoing overhead, such as adding new services, will also be repeated for each partner. Because of the multiple costs associated with trading partners, they're likely to dominate the overall cost for any company with more than a few partners.

Table 1 estimates the cost of partner integration for some hypothetical examples. Even if only two person-weeks of effort are needed to integrate each partner, a large company with 10,000 partners can easily spend \$50 million to \$100 million. Experiences with Electronic Data Interchange (EDI) suggest that actual efforts can take much longer than two person-weeks, thus further increasing the costs. This is the same cost range as the most expensive ERP implementations. Furthermore, such a B2B project is likely to take many years, even with a large integration team. Even for a modest-size Internet trading community, partner integration can cost several million dollars.

The history of EDI illustrates the issues with trading partner management. Companies implementing EDI have typically been able to connect to only 20

Example	# Partners	Est. Person-Years	Est. Cost
Large manufacturer implementing private exchange for suppliers	10,000	415	\$ 62M
New Internet trading community	500	21	\$3.2M

Table 1 — Estimated partner integration costs for a large manufacturer and a new Internet trading community. The table assumes that two person-weeks of effort are required to integrate each new partner, with a fully burdened annual cost per person of \$150K. Actual costs may be much higher, if external system integrators are used.

percent or fewer of their largest trading partners. The chief reason is the incremental costs for adding new connections. Some such costs are related to the Value-Added Networks (VANs) used for EDI; these can be eliminated in a B2B approach based on the Internet and XML. But other costs, such as the need to customize protocols for each trading partner and high testing costs, may well recur in the Internet/XML arena. Are the high costs and sparse coverage of EDI destined to repeat themselves for Internet/XML B2B? Or, can we reduce the cost of trading partner management to enable ubiquitous B2B automation?

Reducing Costs

There's no single technique that will solve all the problems of trading partner management, but with a combination of approaches, it should be possible to substantially reduce the costs. Solutions tend to fall into two categories: those that increase reusability and decrease per-partner custom work, and those that automate or accelerate the per-partner work that remains. Table 2 summarizes the suggestions described in this article.

The first and most important step is to increase the level of standardization, so that the same protocol is used for many different partners. This is more difficult than it seems. For example, in the EDI world, the American National Standards Institute (ANSI) X.12 provides standard formats for various business transactions. Though most EDI installations are based on X.12, they almost always deviate slightly from the standard formats. Typically, each relationship requires a few extra fields that aren't present in X.12. To handle these, the partners find optional fields in the X.12 standard that aren't needed; these fields are then "hijacked" for the extra information. This is confusing and error-prone, since it uses fields in ways that conflict with the documented standard. It also requires different business logic for each relationship.

Unfortunately, similar examples of field hijacking have begun to occur for XML-based B2B. Ironically, the extensible tag mechanism of XML makes hijacking unnecessary. With XML, it's easy to extend a standard by defining new element types for the extra fields instead of misusing existing elements. This lets you develop a single standard set of business logic that handles all the fields used

Reuse protocols and integration code for all your partners.
Don't hijack fields of standard documents for custom information: define additional XML elements.
Design for communities rather than individual partners.
Place partner-specific information name in an address book that is separate from business rules and integration code. Make it easy to define new partner-specific information.
Share your experience with partners and encourage them to use common approaches for their integration.
Offer to provide integration code for partners, which you can maintain in a uniform fashion.
Use cooperative management approaches that allow each of you to observe the state of the other's server.
Use remote debugging to simplify testing and problem tracking.

Table 2 — A summary of techniques for reducing trading partner management costs.

by all partners; there's no need to interpret the same field differently for different partners. Of course, the ultimate goal should be a set of standards complete enough to make extensions unnecessary. In the interim, the XML tag mechanism

Focus on trading communities rather than point-to-point relationships.

can cleanly handle extensions.

One way to encourage standardization is to focus on trading communities rather than point-to-point relationships. If you consider only a single trading partner when designing a protocol, you're likely to end up with a protocol that works only for one partner. If, on the other hand, you assume that various companies will serve each role in the protocol (e.g., multiple buyers and multiple sellers for the same purchasing protocol), you're more likely to end up with a protocol you can reuse. Similarly, you should design your business logic and integration code assuming that it will be used for many different partners.

The next step is to separate partner-specific information from integration code. For example, the code that sends an XML document to a partner as part of a business process should not include the Internet address of the partner. If it does, the code becomes partner-specific

and you'll have to duplicate it with a different address for each partner. This leads to extra work when adding new partners and makes it difficult to change the code, since each change will have to be replicated in all copies.

Instead, you should keep an address book separate from the integration code. The address book should have an entry for each partner that records all information specific to that partner, such as Internet address. Within your integration code, keep track of a "current address book entry" and refer to fields in that entry. This approach separates partner information from information about business processes; it offers several advantages. First, all that's needed to bring a new partner online is to create a new address book entry since existing integration code can be reused. Second, the address book entry provides a clear definition of the information you need to bring a partner online. Third, you need not be a programmer to create new address book entries.

Address book entries can contain more than just Internet addresses; they can contain security information (such as X.509 certificates) and other partner-specific information (such as discount categories or special billing terms). Ideally, the address book should have an extensible format that makes it easy to add new information about partners. If it does, you can handle many partner-specific customizations by adding information to the address book rather than creating custom integration code.

The next step in reducing the cost of trading partner management is to actively participate in development of your partners' B2B integration. Your partners may not have the expertise or motivation to build their own B2B applica-

tions. So, without help, they may never complete the integration. This will be particularly true for smaller partners that are less automated internally. Many of them won't have any prior B2B integration experience. By working with your partners, you can help them prevent "beginners' mistakes" that will cost you both. You can also encourage uniform approaches among your partners, which will reduce your costs.

There are many ways for you to help your partners. The first is to help them install the necessary software and hardware. Most vendors of B2B integration servers offer a special "partner" version of their software at a substantially lower price than the full version. The partner version is intended for the spokes of hub-and-spoke trading communities. This version typically has all the functionality of the full version except that it can communicate only with a single partner. By purchasing partner versions in bulk for all your partners, you can obtain a quantity discount and pass those lower costs on to your partners.

It may even make sense for you to purchase inexpensive server computers and give these to your smaller partners, along with B2B integration software. By doing so, you ensure that all your partners have a minimum level of functionality and communicate with you in a common fashion. The alternative is that your customers will communicate with you, using a much greater variety of protocols and formats, which will drive your costs up. Spending a little money on hardware for your partners may save you a lot of money that would otherwise be spent on fixing problems later.

Another way to help your partners is by sharing integration code with them. If many of your partners use the same back-end applications, the cheapest approach may be for you to develop integration code for those applications and distribute it to your partners. For example, you may find that many of your smaller partners automate only at the level of spreadsheets. You could provide them with integration code that moves data between spreadsheets and XML documents, so all they need to do is use spreadsheets with pre-defined row and column names. This lets you understand and control both ends of the B2B connection.

If a single piece of integration code won't work for all your partners, perhaps you can provide a base package

that they customize locally. Ideally, you and your partners will also be using the same B2B integration server, and the server will support automatic uploading of integration code from your site to your partners. Under these conditions, you can make changes to the business protocol and automatically update your partners' servers. Of course, this approach requires a high level of trust between you and your partners, so it may not be appropriate in all situations.

Carefully consider trading partner management issues before undertaking a B2B project.


Finally, in some situations, it may be possible for you to remotely help manage your partners' integration servers. Some manufacturers of B2B integration servers provide Web-based management tools that can be remotely accessed, using secure Web connections. By giving you remote access to their servers' management tools, your partners can drive down their costs and yours, especially for initial testing and problem tracking. Without remote management access, the only way to test a connection is to talk to your partner via telephone and have them describe what's happening on their end. This is a clumsy and error-prone approach. It's difficult to convey large amounts of information over a phone connection, and information may be misinterpreted (especially if you're dealing with less-experienced partners).

With remote management access, you can directly observe what's happening on both ends. You can query event logs, for example. Similarly, you should consider giving your partners the ability to observe what's happening on your integration server; ideally, they should

be able to observe only transactions that involve them.

Some servers offer remote debugging. This lets you set breakpoints in integration code running on your partners' systems and observe variable values. Even if your partners are not comfortable giving you long-term management access to their servers, it may make sense for them to provide limited access during initial testing — or if serious problems occur.

Conclusion

There's no simple panacea for the challenges associated with trading partner management. Even under the best conditions, it will probably be an expensive part of implementing B2B applications. To keep these costs under control, you should carefully consider trading partner management issues before undertaking a B2B project, and devise a plan for scaling to your entire partner community. Above all, strive for reusability and minimize customization for each partner. This may mean using common protocols, separating partner-specific information from integration code, or installing a standard hardware-software platform for each partner. In addition, take advantage of Internet technology to reduce your costs, whether that means automatically updating your partners' integration code from your code, or using the Web to remotely observe and manipulate your partners' servers. 

About the Author



John K. Ousterhout is chairman and chief technology officer of Ajuba Solutions. He is well known for his work in distributed operating systems, high-performance file systems, scripting languages, and user interfaces. Before founding Ajuba, he was professor of computer science at U.C. Berkeley and a distinguished engineer at Sun Microsystems. He received a bachelor's degree in physics from Yale University and a Ph.D. in computer science from Carnegie Mellon University. Ousterhout is a Fellow of the ACM and has received numerous awards, including the ACM Software System Award, and the ACM Grace Murray Hopper Award. Voice: 650-210-0102; e-Mail: ouster@ajubasolutions.com; Website: www.ajubasolutions.com.