Large Scale B2B Integration With Oracle Fusion Middleware

Sean Carey
Software Architect

Anne Knapp
Vice President of Technology

Introduction

SPS Commerce offers outsourced supply chain services used by more than 12,000 companies. Its services include electronic data interchange (EDI), catalog, barcode labels and online ordering that enable retailers, suppliers and third party providers to eliminate manual order, shipping and invoicing processes – replacing them with automated and integrated operations. Much of the company's growth can be attributed to its integrated EDI Software-as-a-Service (SaaS) offering that experienced a 175% growth rate in 2006 and a catalog service that also grew by more than 145%.

As a hosted solution, SPS' challenge was to accelerate resolution of IT performance issues and enhance operational efficiency and customer service. In order to keep pace with an exponential growth in customers and transaction volume while growing profitably, SPS needed a platform for its SaaS business that could scale, required minimal integration and maintenance overhead, and could be relied upon to deliver a mission critical service to its customers.

Selecting a software provider

SPS Commerce has provided outsourced EDI and business-to-business transaction services to our customers for over ten years using a combination of standard EDI translators and internally developed workflow and integration services. About three years ago, we had reached the point where we felt that this was no longer the best solution to handle our projected growth. The limitations of the system included:

- application and database were not clusterable and therefore difficult to scale with growing demand
- document throughput was limited by batch cycles present in many of the processing steps
- bottlenecks in the system due to inefficient integration and lack of parallelism created data backlogs during heavy load
- the EDI-centric nature of the system made non-EDI trading partner integrations (the fastest growing part of the business) difficult and inefficient
- deficiencies in reporting and audit capabilities of the system made error detection and troubleshooting difficult

In selecting the software platform for our next generation of data center systems, the goal was to address the problems above by purchasing tools and infrastructure which would:

- be highly scalable through clustering at both the database and application tiers
- use event-driven, parallel processing throughout the system for maximum data throughput
- continue to fully support EDI-based integrations while reducing dependence on EDI for data routing and transformation activities
- provide greater visibility to both internal support teams and customers

Following an evaluation of multiple vendors and technologies, the Oracle SOA Suite was selected as the foundation for out next generation data center. In addition to meeting the criteria above, we found that there were additional synergies between the SOA Suite and SPS Commerce's business which made this an especially good fit:

- the principles of Service Oriented Architecture align well with the goals of building and supporting a software-as-a-service business, particularly with respect to business process

automation
- standards-based integration through web services allowed us to rationalize internal systems integration around a consistent set of practices
- from JDeveloper and ADF for user interface development, to BPEL for process automation, Integration B2B for EDI processing and Grid Control, BAM and BI for systems, process and data visibility, Oracle was able to provide a full suite of tools to meet all our needs for building out these new systems
- the move away from EDI as the standard data format provided new opportunities to optimize processing and connectivity for our non-EDI customer base

Moving from design to production

In order to get the maximum value from any SOA implementation, it is critical that sufficient time and thought be invested at the design stage. Understanding the business processes that you intend to model, and how those break down into a set of flexible, reusable services takes time but the return on this investment can be huge in the future enhancement and maintenance of the system. Beyond this, it is important to invest time in the understanding and careful design of inter-process messaging and data structures in order to allow for maximum flexibility. Not only are these two aspects the most important factors in the flexibility of your final design, but they are typically the most difficult things to change once they go to production so it is important to get them right the first time.

At SPS Commerce, our specific problem was to build a system which would support the transformation and routing of data between thousands of trading partners, encompassing hundreds of different document and data formats and business processes in such a way that it could be easily extended and maintained in the future. Obviously, in order to do this we needed to have our systems be as independent as possible from customer-specific implementation details. Also, the business logic had to be dynamic in order to avoid hand-coding and maintaining each of the business processes required by our customers.

The final solution that was reached achieves all of these goals through a multi-layered, configuration-driven approach to data processing. This system is responsible for the transformation and routing of electronic documents such as purchase orders, invoices and shipping notices between thousands of retailers and suppliers. The management and tracking of all of this data is handled by the BPEL Process Manager, with EDI integration handled by Oracle B2B.

1. **Communications**: The first layer handles the communications between the data center and external systems – either hosted internally by SPS Commerce, or resident within a customer's data center. Specialized BPEL processes handle all of the communications with these systems via those systems' native protocols, whether it is web services, AS2, messages queues, FTP or other protocols.

2. **Data Transformation:** The data center processes all documents using a canonical, XML-based format. SPS Commerce utilizes a Java-based transformation service which is capable of mapping between XML and non-XML formats. This service had been previously developed in house, but required only minor modifications to integrate with Oracle Fusion Middleware. This allowed SPS to bring its extensive library of previously developed maps directly into the new SOA solution. Once data has been transformed to the canonical format during the import process it can be routed to any customer, independent of the service and format used by that customer.

3. **Document Routing:** Once a document has been processed through the communications layer and transformed into the canonical data format, it is passed to a routing process. The router is at the heart of the data center and is responsible for processing any customer-specific business rules and routing the document to the recipient. Tracking, correlation and reporting functions are processed at this level as well.

In this system, no complete process is described at the bpel level. Instead, each process in the chain knows how to determine what steps to take in processing the data that it has received, and where to route the data for the next stage. It is important to note also that new logic, processing steps and destinations can be added at each layer simply through changes to the configuration rather than with modifications to code. In this way, as new services are developed they can be deployed and utilized by existing processes simply by describing the function and location of the new services in the trading partner configuration database.

In addition to these primary services, Oracle Fusion Middleware provides essential management services such as an Error Hospital for the collection, reporting and troubleshooting of errors.

Management of the system required additional user interface development for trading partner setup, service configuration, data inspection and monitoring. Here, Oracle's ADF, JDeveloper and Business Intelligence (BI) technologies, along with AJAX, were used to rapidly develop an entire suite of new user interfaces.

Lessons learned

In the course of developing and deploying this system, we encountered a variety of problems which needed to be solved.

The first among these was properly tuning performance at both the application and infrastructure (app server and database) levels. Not surprisingly, it is important to tune the basic application server configuration – settings like memory allocation and connection pooling. More important, though, are tuning for process concurrency in the application server, and for efficient use of the dehydration store. Both cases are heavily influenced by the use of synchronous and asynchronous bpel processes in the overall system, because each type of process has different performance requirements and stresses the application server and database in different ways.

Of these two cases, tuning of the thread pool and process concurrency will probably be the most familiar to those with j2ee development experience. Essentially, the java container that runs the bpel server is configured to allocate a specific number of java threads for process execution. Tuning this value affects the load on the host cpu, but also the load on external services that are being consumed and on database resources. There is a point of diminishing returns where increased concurrency will lead to poorer performance as various resources (memory, disk, cpu, database, etc) are divided among more an more running processes.

The dehydration store, on the other hand, is a database service used by BPEL to store the state of long-running processes to provide high-availability and the ability to recover processes in the event of a crash, as well as allowing calling processes to free up memory and cpu resources while they wait for asynchronous responses from child processes. Overall this is one of the most important features of the BPEL Process Manager, but it does come with a cost in terms of maintenance and configuration. Out of the box, we found that the base configuration of the orabpel database schema was not very efficient, and required tuning in order to eliminate resource contention on a variety of indexes and tables. Of particular concern here were high watermark allocation on many of the blob segments, and index contention at the global cache level in a RAC environment.

In general I believe that there are no easy rules to follow in tuning these components, but I would advise allocating sufficient resources to monitor and adjust app server and database performance as you scale up and out.

As mentioned above, the mix of synchronous and asynchronous processing in your environment will also have an influence on dehydration store application server performance. The basic rule of thumb here is that synchronous processes – where the caller expects a response in the same session as the invocation – typically will be expected to respond quickly and use minimal resources. On the other hand, asynchronous processes – where the caller invokes the child and then closes the session with the expectation that the child will create a new communication session to send back its response – have no specific expectation on response time (and in fact can often be used with processes that may take hours, days or more to respond). The trade-off is in process dehydration and the associated database overhead required to store and retrieve the process state.

Moving up to more business-related challenges, we found that there were a few areas in particular where the EDI world and the SOA world differ in the expectations and where the SOA systems required some intervention in order to improve performance and reliability.

The first example of this was in document size. EDI translators typically do not process entire messages in-memory, and therefore do not usually have a restriction on message size. Consequently, over time various EDI specifications have been developed which model large data sets in a single document – examples include item master data, product forecasts, etc. On the SOA side, we want to model messages – of which business documents are one type – in xml, and render those documents as an object in memory in order to provide query and manipulation capabilities through xpath, xsd and other technologies. In order to resolve this conflict, we have found that there are a variety of approaches which can be applied in different situations.

One approach is to manage data size, either by working with trading partners to break up large documents into smaller pieces or by using preprocessing to achieve a similar result within one's own system. This requires more work up-front but is, in the long run, the least intrusive from a systems perspective. Another approach is to store large messages out-of-band. This is especially effective if some form of filtering can be applied up-front in order to extract the “interesting” part of the message. This way the integrity of the overall messsage is maintained without incurring the overhead of passing the entire message throughout the process. The down-side here is that transformations or other global message operations may still have a problem dealing with the message as a whole, and processes have to be modified to have awareness of out-of-band data and how to retrieve it if necessary.

Another example of EDI/SOA disconnect is in message bursting. Traditional EDI and/or B2B messaging was often managed through batch cycles, scheduled to run on specific intervals throughout the day. With this approach, messages are accumulated over time and transmitted in large groups. With a SOA system, we are typically setup to try to handle all incoming requests simultaneously rather than sequentially. Message bursts can be problematic because they create high-load conditions in the system, which degrade performance for all users.

As in the large document case, working with trading partners to explain the problems of message bursts and how better results can be achieved through event-driven processing or smaller, more frequent batches of messages often can produce the best results for both parties. Failing this, we have found traffic shaping to be an effective way to deal with the problem. This is more expensive up front because it requires that batches of data be torn apart into individual messages which can then be trickled through the system on a timely basis, but the positive impact on systems performance has proven to be worth the investment.

Project Results

The migration to SOA and Oracle Fusion Middleware has enabled SPS Commerce to increase its capacity for growth and change and provide fast, reliable service to its customers. By choosing Fusion Middleware, a system that is central to every aspect of SPS Commerce's offering, we have experienced the following short and long-term benefits:

- Allowed the company to confidently process 3 million transactions monthly and scale to meet increasing demand
- Reduced processing time for individual documents by 80%
- Improved error handling, notifications, and document-tracking capabilities
- Minimized the workload for the tech support team by helping them easily identify problems and resolve issues
- Automated processes to reduce manual labor
- SPS Commerce is now a more agile platform for B2B integration