SaaS Integration: Build vs. Buy?

By: Bob Moul, General Manager, Dell Boomi

It seems seductively simple. You’ve just purchased a SaaS application and you need to integrate it with one or more of your core systems. Why not just build the integration yourself? Bang out some code, do a little testing and you’re up and running right? Unfortunately, what seems simple on the surface is only the tip of the iceberg. And it’s a very large and complex iceberg to boot.

As more and more businesses adopt SaaS technology, they are faced with making a critical decision between developing custom integration or purchasing integration-as-a-service. Forward thinking businesses do not plan integration projects piecemeal — they look at the big picture including back- and front-office applications as well as their supply chain. In addition to SaaS applications, many businesses are supported by a complex ecosystem consisting of a combination of on-premise, platform-as-a-service (PaaS), e-commerce, and cloud-based applications. Rather than look at each integration project in a silo, best practice is to select an integration strategy that will support all of the above in a single, seamless solution.

According to Benoit Lheureux, an analyst for Gartner, “the challenge for customers is to know when to choose one approach over another. The answer depends heavily on each customer’s particular situation, including factors such as internal integration skills and overall B2B strategies.”

Challenges with Custom Coding Integration

For those businesses considering building custom integrations, there are several key components that must be included in order to develop a complete integration solution: the integration process itself; navigation of the application programming interface (API) (including learning the proprietary API’s of the applications being integrated in order to build “connectors” to extract and load data); monitoring tools to provide logging to simplify error resolution; redundancy mechanisms to automatically handle scenarios where the applications being integrated become unavailable; and resiliency to support the frequent release cycles of SaaS applications and their corresponding APIs. Most people only account for the integration process itself, overlooking the other critical functionality and wind up spending exorbitant time maintaining their custom code.

API Fluidity

If a separate In the dynamic SaaS ecosystem, APIs are updated on average 4–12 times per year. The more mature SaaS applications will release 4 times a year and will ensure backwards compatibility.

“We use Boomi to provide the critical link between our customer-facing web services and Salesforce and the value to our company has been through the roof! The implementation was a fraction of the cost of custom coding, reports that used to take a day to generate are now done in five minutes.”

Tom Fox, IT Project Manager
Global Forex Trading
This means that, generally speaking, the integration will continue to run as new versions of the API are released. However for the vast majority of SaaS applications, the releases occur much more frequently and in many cases are not backwards compatible. This means that the integration code must be modified in-step with the API change in order to ensure the integration runs unaffected.

API Customization

As changes are made in the SaaS application by the administrator, such as adding a custom field, that new field will instantly be made available in the API. In most cases, the integration will need to include this new custom field so that it can be added into the code that handles the transformation of the data from one application format to another.

API Multi-Tenancy

When coding to multiple tenants of a SaaS application, you will need to maintain code sets for each tenant, or architect for multi-tenancy up front. This is necessary because the vast majority of SaaS applications allow point-and-click customization of their application, which means the vast majority of customers will in fact customize theirs, and each one who does will have a schema that is unique to their tenant. The integration process built will therefore need to implement the meta data APIs that the SaaS application provides on top of the standard data access APIs.

Communication Resiliency

Integrating with SaaS applications should assume that the natural “ebbs and flows” of the internet are factored into design. API calls will inevitably fail or time out, and this is nothing more than internet latency. Resiliency must be coded from the ground up so that connections to APIs are automatically retried before signaling error and forcing human intervention.

Error Handling

A reporting system must be developed in order to collect, display and manage log information as the integration processes data. This system also needs to be able to broadcast errors to interested parties for provide proactive notification of issues.

The primary uses of this information are:

- Troubleshooting in response to errors — The goal here is to quickly pinpoint failures of a particular execution, and then details of what happened for that execution. Typically, the integration will consist of multiple steps that include connectivity, data transformation, business logic, duplicate checking, content based routing, etc... and so it is critical that you can quickly determine which of these steps failed.

- Developing an audit trail to confirm delivery of data — This is particularly important when communicating data outside the enterprise, such as to remote divisions or business partners. The goal is to provide easy ways to search for execution history of a given integration by allowing for searching of meaningful fields within the data, such as a number that uniquely identifies a customer.

Disparate programming languages and API standards

Depending on the application being integrated, a number of APIs require use of a proprietary SDK to code against their API. The resources required are then compounded when the other applications being integrated force the same approach. The developer ends up having to learn multiple API standards, technologies and potentially programming languages as well. Even “standardized” APIs based on technologies such as SOAP/WSDL require a significant effort in order to learn the API, underlying data model, authentication mechanism, and query capabilities.

Business Logic

Beyond connecting to the applications to be integrated and coding the transformation to convert the data from one format to another, there is almost always business logic required to be included into the integration. This extends the life of the coding project, and is another area that will be fluid as the business process gets sorted out, require multiple iterations for code to be written and maintained.

Performance Optimization

When integrating with SaaS applications, each call against the application is transmitted via a secure internet connection, typically over HTTPS. When attempting to code against the raw API directly, there are important considerations around the nature and frequency of how the API is invoked. Done wrong, this can have order of magnitude type effects on performance. Yet relying on a WSDL will not offer any guidance on what to avoid or provide any best practices for scalable integration with the SaaS application.
Custom coding integration processes is generally considered to be expensive, time-consuming and a drain on internal resources. Some find it to be a desirable option when a business needs to quickly connect no more than two applications, the data value is low, and it is not expected that the applications will change. However, given the dynamic nature of SaaS and cloud-based applications and the high value of the data exchanged between them — that scenario is rare.

**Integration-as-a-Service**

Forward thinking companies that have realized the outstanding value proposition of the SaaS model and are looking for IT infrastructure and support that offers the same. End users research, try, and purchase SaaS applications in a self-service manner without ever leaving their web browser. Following purchase, maintenance is low as there are no servers to install or maintain and updates are handled centrally by the SaaS provider. Savvy businesses are seeking integration solutions built from the ground up as pure SaaS which also offer the ability to build, deploy, and manage integration processes from a web browser.

The recent development of Integration as a Service is a natural outcome of the convergence of Service Oriented Architecture (SOA) and SaaS. As businesses of all sizes migrate to SaaS and cloud-based applications, the need for solutions to allow them to interoperate and exchange data is obvious — as is the call for such solutions to be deployed and purchased in the same on-demand model.

When developed and built from the ground up in pure SaaS technology¹, integration solutions are cost-effective, scalable, and flexible. Businesses minimize the use of internal IT resources as the service is typically made available in a completely self-service model, and can be configured, deployed and managed right from the web browser without having to write code or install any software or hardware on-premise. For organizations with multiple business units, these solutions can be deployed to multiple geographic locations from a centrally managed, web-based dashboard. CIOs can also gain a comprehensive view of all integration processes within their organization with this same dashboard. This is a growing challenge as individual departments purchase SaaS application subscriptions often independently from one another.

Best of breed integration service providers offer a single, seamless solution for a business’ entire application portfolio including on-premise, cloud-based, and SaaS environments. Many providers offer pre-built connectors to leading applications as well as the ability to develop custom integration processes quickly and easily via visual, drag-and-drop technology. As mentioned previously, integration service providers should offer a secure method of data transmission both in the cloud and behind the company’s firewall for on-premise systems.

**Conclusion**

In summary, the advent of Software as a Service and Cloud Computing has revolutionized the software industry by providing access to enterprise-grade software and services via the web to businesses of all sizes. SaaS and cloud environments are characterized by web-based delivery, multi-tenancy, and centralized management and updates — completely unlike traditional software. As a result, new infrastructure and supporting services, such as on-demand integration, are crucial to the success of this model.

In choosing an integration strategy, businesses should take a strategic, long-term view and avoid short-term decisions that could result in an ongoing drain of valuable IT resources and exponential costs to the organization. It is equally critical that businesses consider the need for scalability — both in terms of the growth of the customer base and the expansion of back-office solutions to include future purchases of SaaS, PaaS, and cloud computing applications. Best-of-breed integration solutions will mirror the SaaS value proposition and allow for scalability and expansion as businesses grow and change over time.

**About Bob Moul**

Bob Moul is General Manager of Dell Boomi. Bob joined Boomi in late 2005 as a member of the board of directors and was appointed president and CEO in August 2006. He has led the company’s successful transformation from a traditional software product company to a leading infrastructure provider in the SaaS and cloud computing industry. Boomi is the creator of AtomSphere, the industry’s #1 integration cloud.

¹ Buyer beware of on-premise software vendors who market themselves as SaaS by hosting their packaged software product in a data center. That is an ASP model not SaaS and it will not scale over time, forcing their customers to absorb the increasing costs.