

# Making Sense of SOA, SOAP, and REST for Enterprise Data Integration

A SnapLogic White Paper

November 2007



SnapLogic, Inc.  
1700 S. Amphlett Blvd., Suite #221  
San Mateo, CA 94402  
[www.snaplogic.com](http://www.snaplogic.com)  
(650) 655-7200

# The Clamor around SOA, SOAP, and REST

A couple of lively debates are roiling the world of enterprise IT, sweeping up everyone from CIOs and system architects to development teams, security officers, and network administrators. The first debate involves the pragmatic value of a Service-Oriented Architecture (SOA). The second is a dispute between the advocates of SOAP-based Web Services and the advocates of Representational State Transfer (REST)-based Web services to determine which architectural style is best suited to meet the objectives of the agile enterprise.

## The SOA Debate

For most vendors and analysts, and certainly for many enterprises, SOA connotes an enterprise-wide initiative to replace monolithic, siloed applications with a collection of broadly accessible *services*. Vendors, analysts, and architects have been promoting the value of SOA for years, touting the long-term advantage of investing in small, reusable components that can be rapidly assembled to create new services, rather than continuing to rely on lengthy development cycles and large, complex, and cumbersome applications.

SOA has certainly gained momentum in the world of enterprise IT. Recent polls of CIOs show SOA and Web services shooting to the top of the priority list for most enterprises, in many cases bypassing traditional front-runners like security and compliance. Other top priorities, these polls show, include Business Process Management (BPM, the management and optimization of business processes, which are increasingly Web-enabled) and Business Intelligence (BI, the benchmarking and analysis of operational performance). The combination of SOA and BPM makes sense. SOA is about creating an IT infrastructure that can be flexibly deployed to address changing requirements for business operations and business services. SOA, BPM, and BI all have to do with agility, efficiency, and visibility.

Given the ever-increasing tempo of the global economy, businesses have a keen interest in becoming more agile. One way to do this is by replacing costly, monolithic enterprise applications, which usually take years to develop and deploy, with a collection of smaller, independent business services that make business logic (e.g., authentication processes, logistics services) available on demand. Over the medium and long term, an architecture that stresses re-usable components will always prove more cost-effective and flexible than an architecture that relies on one-off applications to perform business services.

Nonetheless, some business managers, IT managers, and programmers are rightfully skeptical of the hype that has built up around SOA. SOA implementations take time—usually years. Everyone from SOA vendors to SOA adopters is on a learning curve, trying to figure out what really works and what doesn't in a SOA deployment. Progress is being made, but most organizations are still in the pilot phase. So far, the results of current implementations have been spotty. A recent study by Nucleus Research found that, so far, only 37% of SOA implementations have delivered a positive ROI.<sup>1</sup>

This leads some skeptics to say that SOA is too grandiose, comprehensive, and impractical a vision to meet the pressing needs of business and IT. As the analyst firm ZapThink notes:

Most often, the single cause of failure for SOA is inappropriate scoping of the SOA project. Companies too often seek to make SOA an enterprise-wide effort, even though the business case for that is typically not justified. . . . SOA is simply not appropriate for all problems.<sup>2</sup>

This perspective, coming from a firm known for its bullish attitude on SOAs, reflects a growing consensus that while SOAs have their value, enterprises would be misguided to try to funnel all their integration and Web services solutions through a grand SOA implementation. SOA should not be all or nothing. Enterprises can take an incremental approach to services roll-outs, sometimes building on SOA and other times, when appropriate, deploying distributed data services around the edge of the IT infrastructure.

---

<sup>1</sup> <http://www.sdtimes.com/article/column-20071001-01.html>

<sup>2</sup> ZapFlash Newsletter, "What's Not Killing SOA?", October 16, 2007.

## SOAP vs. REST

Another debate focuses on a closely related topic: Web services, the delivery of application and data services using HTTP and other Web-based technologies.

Web services provide a convenient and powerful interface between applications. This is, in part, because Web browsers and Web servers are now ubiquitous, and it makes sense to leverage this familiar client/server infrastructure for new services. It's also because Web services have the potential to deliver automated and ad hoc services that span multiple enterprises and organizational domains. And finally, it's partly due to customers demanding that new applications and services be able to integrate with products from other vendors.

The promise of SOAs based on Web services drove the creation of the SOAP and WS-\* standards and for a long while it seemed that SOA meant SOAP and WS-\*, and the terms became nearly synonymous. Meanwhile on the public Internet, different kinds of Web Services such as the data access APIs provided by sites like Flickr, Google, Amazon, Yahoo and the data available from other sources like RSS feeds—all based on simple RESTful interactions—were being rapidly adopted for a wide range of loosely coupled, integrations.

The success of simple, public RESTful Web services caused some developers to reconsider using the increasingly complex WS-\* and SOAP to achieve their SOA objectives. They then began to promote REST for SOA<sup>3</sup> as a credible alternative that could be successful without any of the overhead or complexity of SOAP or WS-\*. Therein lies the debate: SOAP vs. REST.

---

<sup>3</sup> This has recently been given a name Resource Oriented Architecture, or ROA

Web services developers seem to be confronted with a choice: either develop services based on SOAP interfaces, the WS-\* family of specification, or develop services based on a more bare-bones REST infrastructure based primarily on a few basic HTTP commands (GET, PUT, POST, DELETE). The SOAP and WS-\* specifications were developed to provide reliable, secure messaging for mission-critical applications. But they add complexity and overhead to services. In contrast, programming in a REST style is simpler and faster. It omits some of the rigorous controls available with SOAP, but often these controls are not needed for internal services, or the same level of control and security can be achieved through the standard techniques developed for VPNs and security of Web traffic.

Given the business need for more information from more sources, it's understandable that some business managers and IT teams would want to take advantage of the rapid development cycles enabled by REST. The promise of direct access to services, via sophisticated, interactive interfaces made possible by AJAX and Web 2.0 programming techniques, is too attractive to pass by. Analyst firms, who formerly promoted best practices based on SOAP and WSDLs, now concede that IT organizations have been right to prefer REST for most of their Web services solutions. Anne Thomas Manes of the Burton Group, for example, states flatly that the future of Web services is REST.<sup>4</sup>

But many SOAP supporters are alarmed by the growing popularity of REST. They worry that REST's popularity comes at the expense of the more rigorous programmatic discipline derived from the SOAP and WS-\* standards, and that REST is undermining the primary objectives of SOA: Instead of contributing services to a strategically important, tested, and sanctioned collection of enterprise services, REST developers seem to be creating "quick-and-dirty" Web solutions, many of which may turn out to be unique, un-reusable programs—as much a "one-off" as the monolithic enterprise applications they are designed to replace.

---

<sup>4</sup> "Burton sees the future of SOA and it is REST," Rich Seeley, SearchWebServices.com, 30 May 2007.

How should one make sense of this debate? Even in organizations that support the vision of SOA and are pursuing SOA implementations, how should programmers meet their urgent, short-term deadlines (e.g., the data feed needed for the BI dashboard, which is just as high priority as the SOA initiative) until the comprehensive roll-out is complete? Is there room for a middle way, an approach that acknowledges the value of structured SOA based on SOAP and WS-\*, while enabling developers to solve problems efficiently in the here and now?

## **The Router Analogy**

One way of understanding the best use of SOA, SOAP, and REST is to compare them to other areas of IT infrastructure where the requirements are pervasive, yet still vary tremendously across the enterprise. One such area is network infrastructure. Today, no one would disagree that a network needs both switches and routers. But when switches first appeared there were people that considered them a throwback to the earlier days of bridged networks. Even among routers, there are two fundamentally different types designed for completely different needs, yet everyone today would agree that each has its place. Core routers route packets between other routers within the core of the network, while edge routers route packets from the core of the network to other networks. No IT organization would confuse the two types of routers. They serve different purposes, and many enterprise networks needs both types.

But that doesn't mean that *every* enterprise network needs *both* kinds of routers. Department LANs, small offices, and home offices typically require only edge routers. Conversely, a large telecommunications company might have an entire division that runs exclusively on core routers, using no edge routers at all.

It's clear that a services architecture shares many of these network characteristics. At its core, the architecture requires an essential set of stable, high performance services. These are often implemented with an Enterprise Services Bus (ESB). The ESB provides messaging and transactional services for tightly coupled integration with applications. It's a centralized function, like the function of a core router. But, analogous to a network "edge," there's also a "services edge" that supports varied, loosely coupled integrations with endpoints such as Web sites, desktops, departmental applications, hosted applications, public Web sites, search and mapping services, RSS feeds, etc.

Does every integration require an investment in a SOAP/WS-\* based SOA? No. These SOAs are like core routers. They're great for certain operations, but they're not appropriate for all operations. Other integrations, which take place at the functional edge of the enterprise infrastructure, can take advantage of technology that lends itself to more rapid and flexible development and deployment.

## **The Right Solution for the Right Job**

Just as edge and core routers have their place in the IT infrastructure, REST and WS-\* have their places as well. RESTful, lightweight services are appropriate for many data integration applications, especially those with tight deadlines and scarce resources. The combination of REST and dynamic languages makes it possible for an IT engineer to quickly, efficiently extract data from a source, groom it, massage it, and transmit it to some other target on the Edge—whatever that target might be: a file, a department database, a Web site, a browser, etc.

For example, to follow up on a marketing campaign while the leads are still hot, a regional sales manager might want to export a list of names from sales forecasts tracked by Salesforce.com and compare that list to a spreadsheet listing names and contract information culled from the campaign. Such ad hoc investigations are ideally handled using local resources, Web access, and a dynamic language like Python or Ruby, rather than a centrally managed SOA implementation that is midway through a 9-month roll-out. The sales manager wants the data now. Edge data integrations can deliver data that quickly.

## Edge Integrations for the Agile Enterprise

It's important to stress that, even though edge data integrations do not make use of major pieces of SOA infrastructure, such as ESBs, they still uphold some of the key principles underlying SOA implementations. *Edge data integrations complement and enhance SOA implementations, providing essential data service capabilities for tactical initiatives and daily operations.* In today's fast-paced, global economy, agility and efficiency are essential for every organization. IT departments can go a long way toward achieving agility and efficiency objectives by exploiting the capabilities of edge integrations.

One common way to think about data integrations is to consider them to be *pipelines* that begin with reading a data source, transforming it in some meaningful way, and delivering it to its destination. Business processes as varied and vital as Business Intelligence, reporting, content publishing, and even asset management all rely, in one way or another, on these kinds of data integration pipelines. Business needs in all these areas change quickly. It's unreasonable (and sometimes impossible) for IT departments to wait for their organization's SOA initiative to catch up with the latest business or technical requirements of a given department or a given group of departments. In the same way that the Web itself evolves, IT engineers can implement pipelines at the Edge, using REST-based communications and taking advantage of dynamic, high-level programming languages such as Python.

Compared to multi-year SOA projects, such implementations will appear lightning fast and tactically expedient. Like Core SOA projects, Edge implementations can apply key SOA principles—callable services, re-usable components, audit trails and governance, Web-based communications—to produce key SOA benefits—agility and efficiency. Here's how they do so:

### **Agility:**

- Dynamic languages enable engineers to quickly program integrations, enabling them in some case to create custom solutions in a matter of minutes or hours.
- Programs written in dynamic languages are easy to understand and modify.

- Edge integrations make data available as a service, giving business managers ready access to information to analyzing situations and making decisions. Edge integrations enable managers to manage quickly and nimbly

**Efficiency:**

- When used in a modular, building block framework with a pipeline repository, Edge integrations can adhere to the same principle of component re-use that larger ESB-based implementations do. The difference is that these components can be created and deployed quickly, with minimal technical overhead.
- Since Edge integrations make data available as a service, as more edge integrations are deployed, each new integration becomes simpler, faster to implement and easier to manage. Operational efficiency increases. So does business agility.

In summary, it's useful to think of edge integration frameworks as a service-based solution for use at the department or division level. Like larger, centralized Core SOA projects, Edge integrations strive to deliver agility and efficiency. But they do so with tools that are readily available—dynamic languages and REST—and they can be implemented easily, without requiring costly server upgrades or major middleware infrastructure deployments.

What features and capabilities should an Edge integration solution offer to meet the rapidly changing demands at the Edge without compromising the vision, best practices, and IT investments associated with the Core?

- Graphical tools and high-level languages to support rapid development
- Data connectors for the most popular business applications—the likely sources and destinations for data in data pipelines
- An environment for rapidly create new data connectors, so that development is never hamstrung by the lack of a certain connector
- Flexible endpoints; that is, endpoints that can be easily configured and accessed, without requiring complex infrastructure or programming

- Access controls and security, so that ease-of-integration never compromises data security and policy compliance
- Data lineage controls, so that the source, age, and authenticity of data can always be verified.
- A programming environment that supports rapid development

The remainder of this paper will examine SnapLogic's Open Source solution for Edge integrations.

## **The SnapLogic Data Integration Framework**

SnapLogic™ is an Open Source data integration framework that uses the universal standards of the Web and applies them to the problems of data integration at the Edge.

Unlike commercial integration solutions for the Core that are designed for complex, tightly coupled integration projects, SnapLogic has the power and flexibility to address a wide range of data integration requirements. The SnapLogic data integration framework is ideally suited for the “Last Mile” of data integration, ensuring that departments and divisions have access to the data they need, when they need it.

The SnapLogic Data Integration Framework offers:

- SnapLogic Designer, an easy to use, browser-based drag-and-drop interface for combining data integration SnapLogic Resources into SnapLogic Pipelines and for running Pipelines.
- A full programmatic interface in a dynamic language, Python, for writing and customizing data integration Pipelines.
- A command line interface, enabling SnapLogic Pipelines to be run through shell commands, cron jobs, and other interfaces.
- A metadata repository for storing Resources and Pipelines for re-use.
- A growing collection of free, Open Source Connectors and Pipelines for common data sources such as Apache, Oracle, QuickBooks, Salesforce.com, etc.

Like the Web itself, SnapLogic standardizes the access protocols, access methods and data structure for every data interaction. These RESTful interactions allow data to be exchanged between servers in an organized way that makes it easy for the data to be processed and interpreted. SnapLogic creates a resource-oriented data services layer that provides transformation and integration services for edge integrations. Through SnapLogic, users can readily access the data they need for their jobs—whether that data resides in a database, a Web site, a file, a hosted application, or some combination of these sources.

For more information about the SnapLogic solution or to download the latest SnapLogic framework, please visit [www.snaplogic.com](http://www.snaplogic.com).