WHITE PAPER

Give your SOA a REST with APIs:

The cornerstone of a modern digital business foundation

01

INTRODUCTION

In the early 2000s, the Services-oriented Architecture (SOA) paradigm was heralded for its agility potential. However, its loose coupling and use of standards did not widely translate into the kind of business transformation that adopters envisioned. The technology proved to be more complex and challenging to manage than experts had predicted. In contrast, the RESTful API, an evolutionary step up from SOAP, has proved to be adept at delivering flexible application integration at a skill level applicable to widespread adoption. It enables the true digital transformation of customer experiences and business itself.

Now, organizations that embraced SOA can transition to an API-based architecture and benefit from its improvements in connectivity and manageability.

While the distance from SOA (and legacy applications) to APIs may seem like a chasm, with the right practices and platform, the path from SOA to API is a reasonable undertaking. This paper reviews how to make the move and convert SOAP web services into a productive part of a broader API ecosystem.

SOA: A revolutionary idea that stalled

As the Internet grew in popularity in the 1990s, demonstrating the unprecedented market power of open standards like HTML and HTTP, people began to wonder if the same effect could be achieved with software itself. Could applications leverage open standards to break down barriers to integration and inter-operation in the same way that web standards had rocketed the Internet out of the lab and into the economy?

How hard would it be, software architects wondered, to establish software that could invoke functions from other applications just as easily as consumers could jump from one website to another? For example, how about a consumer financial website that could tap into multiple exchange ticker feeds — without the costly, time-consuming custom coding required at the time? Up to that point, connecting applications had involved the use of proprietary middleware and custom-coded links that required specific firewall configurations to work.

At the start of the new millennium, the biggest tech companies took the industry by surprise by agreeing to work together on a shared set of open standards that would allow for free, Internet-based integration of applications. These were the “Services-oriented Architecture” or SOA standards. Based on the XML language, they comprised Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and several others. These standards formed the basis for significant new platforms such as Microsoft .NET, BEA (later Oracle) WebLogic, the new generation of IBM WebSphere servers, SAP NetWeaver, and Oracle Fusion Middleware.

The SOA concept was viable. With all the big players on board, many enterprises embarked on ambitious programs to build Services-oriented Architectures. This effort involved making software functionality available as SOAP-based web services that could easily connect, via HTTP, with SOAP web services both inside and outside the enterprise. New constructs like the “Enterprise Service Bus” (ESB) emerged, enabling organizations to place their web services in software structures that were easy to integrate into new applications.

SOA worked, up to a point. Many successes offered users a level of convenience seldom experienced before. Major applications for ERP and logistics released new editions with hundreds of functions available as web services. Developers discovered a new agility as they linked applications and created new user experiences. New "composite apps" built using web services enabled greater customization to user requirements.

The SOA experiment began to falter after a few years, however. While revolutionary for its openness and the unusual degree of cooperation between competing tech vendors in its use, SOA was still a cumbersome technology. ESBs were expensive and difficult to maintain. The assumption was that business value would come when the SOA project was finished, but the work never seemed to end. The business value sought from well-organized SOAs, with intuitive directories of web services that could be reused across multiple applications, was quite difficult to achieve.

Instead, what evolved during the first decade of the 21st century was a decidedly mixed environment. It included some organized SOA programs, a fair number of ESB-oriented architectures, many enterprises with scattered, disorganized web services, and more than a few that still lagged far behind the SOA curve. A new, simpler approach was needed.

Digital transformation = customer experience transformation

SOA was, in some ways, a victim of its success. Flawed as it was, it demonstrated conclusively that a massive economic prize would come to those who could transform their businesses through technology. SOA opened the eyes of business leaders to the potential of composite applications that could include services relevant to the user's role. They saw the promise of loosely coupled applications to create radically new, better customer experiences. The race was on to develop the technology that would enable such a transition.

Omnichannel customer engagement offers an example of digital business transformation in action. In a bank, for instance, it could be possible to connect the customer seamlessly to account information, transactions, third parties like investment accounts, loan services, and customer service representatives via phone, PC, smartphone, in-branch kiosks, and so forth. To do this, a sprawling collection of applications would have to find simple, agile connections. SOA was too clunky for this vision.

The solution appeared in the form of standards-based Application Programming Interfaces (APIs). The API, a software entity that connected applications and hardware, was far from new; but with the addition of simple, open standards, it became a vehicle of a true IT revolution. When IT literature discusses APIs today, it invariably refers to APIs built for the Representational State Transfer (REST) message protocol and the JSON programming language. Such RESTful APIs, as they're known, offer developers and architects the kind of ultra-simple connectivity envisioned, but never realized, by the SOA standards.

RESTful APIs, using network protocols like HTTP, can link virtually any two pieces of software with remarkable ease. The agreed-upon standards for RESTful APIs contain a handful of simple methods like GET and PUT (data).

In the bank example, as long as developers had the APIs' addresses (URLs) and data parameters, they could assemble a consumer banking app that gave the customer access to accounts, trading systems, and so forth with relatively little effort. Also, it was pretty simple to change the app and its underlying API connections. In this way, APIs became the tools of digital transformation.

The differences between SOA and APIs

APIs seek to accomplish similar benefits, but, unlike SOA, their core design ethos is simplicity. Their simplicity was critical in enabling mobile devices to interact with other systems. SOA was never designed with this kind of simplicity, so SOA's use was limited outside of the enterprise, e.g. with partners and so forth. For example, with regard to resources, the API is addressable by a URL using the standard web technology of HTTP. The API approach offers the ability to start small. From limited initial functionality, it's easy to grow quickly into more sophisticated use cases.

Moving to API-driven customer experiences

The "lean startup" motto of "fail fast" is very apt for the process of embarking on an API program. In some ways, you are like a startup in the midst of the business. You will be developing technologies that unite previously separate systems and business processes in novel ways—just like a startup. You might be a bit of a disruptor as well, with all the good and bad that brings.

Start small. Fail fast.

You don't have to launch a super-sized enterprise API initiative. In fact, that can be a route to real trouble. It's better to start from business goals and think outward from there to deliver value quickly.

Start with the customer. Today's digital business interactions include an ever-increasing set of touchpoints. Think about how the customer experience might benefit from going across these touchpoints.

For example, can a customer interaction move from desktop to tablet or phone and back without adding inconvenience? If not, that's an excellent place to start thinking about APIs and how they can facilitate customer experience across touchpoints. API technology is the foundation for connecting systems and touchpoints. It also connects the people at the heart of customer experience.

Overview of the process

Once you have identified a customer experience that can be improved by APIs, it's time to begin. The first step is to model the process flow to understand what will happen at each step in the customer's interactions with various underlying systems. This flow can be a bit complex, given how digital businesses operate.

For example, the first customer touchpoint might be through a web browser that hits an e-commerce engine. The next touchpoint could be through a mobile app that pulls the transaction status from a fulfillment and logistics application. A third touchpoint could be over the phone and an Interactive Voice Response (IVR) system that pulls order status data from a freight handling application at a third party, such as the U.S. Postal Service.

As ***Figure 1*** shows, each touch point (and its associated application) needs to be made available as an API. Some of these application functions might already be available as SOAP web service. In that case, it will be necessary to do what's called SOAP-to-API conversion. Many API management tools provide this process natively. It renders the SOAP web service into a RESTful API.

Each API needs an "API consumer," which is typically a piece of code on the customer side. In our example, the mobile order tracking app needs to "consume" the API exposed on the fulfillment application.

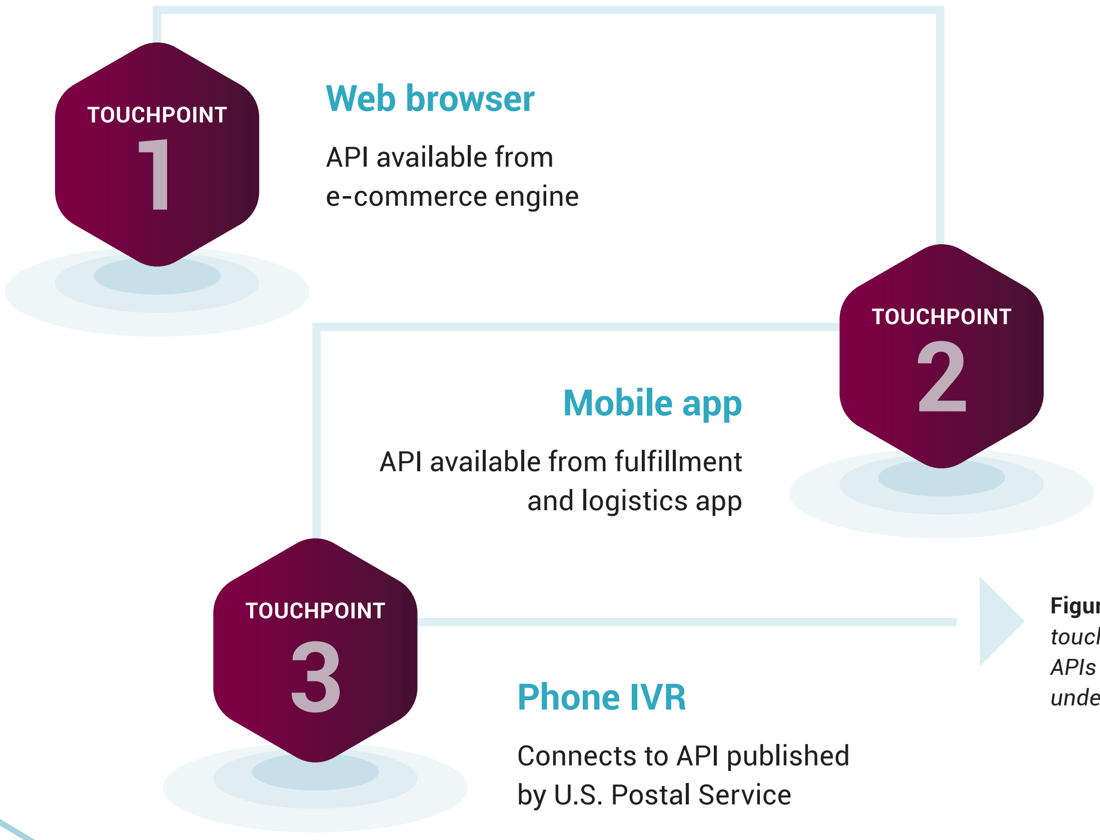
TOUCHPOINT 1
Web browser
API available from e-commerce engine
TOUCHPOINT 2
Mobile app
API available from fulfillment and logistics app
TOUCHPOINT 3
Phone IVR
Connects to API published by U.S. Postal Service

Figure 1: *Multiple touchpoints may require APIs exposed on multiple underlying systems.*

5 API best practices

Moving from SOAP web services and legacy applications to API-driven customer experiences works better when your organization follows proven best practices. You need detailed planning, executive buy-in, budgets, accountability, agreed-upon tasks, calendars and so forth. That's a given. If you don't have these kinds of basic project management practices, you'll get into trouble with APIs.

Many best practices have emerged in recent years that make the SOA-to-API transition process work as well as possible. Highlights include:

01
Mapping customer behavior

For APIs to be relevant, they must align with and be able to adapt to changing customer requirements. This might mean anticipating additional information valuable to the customer, for instance. Understanding browsing patterns is critical.

02
Understanding the difference in use of web services and APIs

The implementation of SOA often resulted in hundreds, if not thousands, of web services according to the granularity of design. APIs, by design, should make the task of the developer who uses them easier and this often results in combining web services in a single API call. For example, a developer writing an app does not care about the many steps it may take to enter a sale or a new contact. One API method that uses as many web services as necessary to accomplish the task helps to simplify the developer's task.

03
Planning how APIs will move from isolated teams to organization-wide embrace

If you do APIs right, you will naturally evolve past the "startup" phase. A time will come when you will want to enable as many people in the organization to use APIs as possible. APIs are, after all, a tool for the business as much (or more) than they are part of IT. It's a good practice to plan for their spread across the organization.

04
Managing the control of APIs throughout the organization

Spreading the use of APIs should not mean losing control. Strong controls over APIs are essential to their effective adoption in multiple projects across different business units. Maintain unified aspects of your APIs for greater efficiency. The best approach to achieving control is by adopting an enterprise-wide API platform.

05
Making the development of APIs more accessible to a larger development pool

Successful API programs include an API developer portal. The portal provides the ability to leverage your initial team of API experts to enable other people in the organization— both API builders and consumers. Part of this involves providing a catalog of existing APIs. With this capability, developers can test existing APIs to learn how to use them.

Community forums provide a place for peer-to-peer learning and so forth. It's surprising to see how people will want to use APIs once they understand their potential. It's not unusual to see unplanned innovation occur as technical people outside of IT use existing APIs to build tools that address their business challenges.

The role of the API management platform

An API management platform gives the organization the means to streamline the API lifecycle, from build to retirement. You can perform mediation, like the SOAP-to-API conversion that simplifies building APIs from your existing services. The platform helps accelerate API use with reliability, security, and scale. From planning through development, deployment, and retirement, it makes it possible to put APIs to consistent use to improve the customer experience.

Originally a simple construct, API management continues to evolve to keep up with the complexity and scale of increasingly sophisticated API strategies.

In fact, the new generation of platforms often does more than just manage the API lifecycle. As ***Figure 2*** suggests, platforms can be quite multi-faceted. For instance, they now routinely handle microservices in addition to RESTful APIs.

They help with the use of APIs and microservices in application development, B2B use cases, and content collaboration. For example, they can facilitate the process of making APIs work between partners. They can also enable advanced analytics of API and microservice performance and dependencies.

Given the ability of APIs to streamline integration, API management is often a central component of a hybrid integration platform.

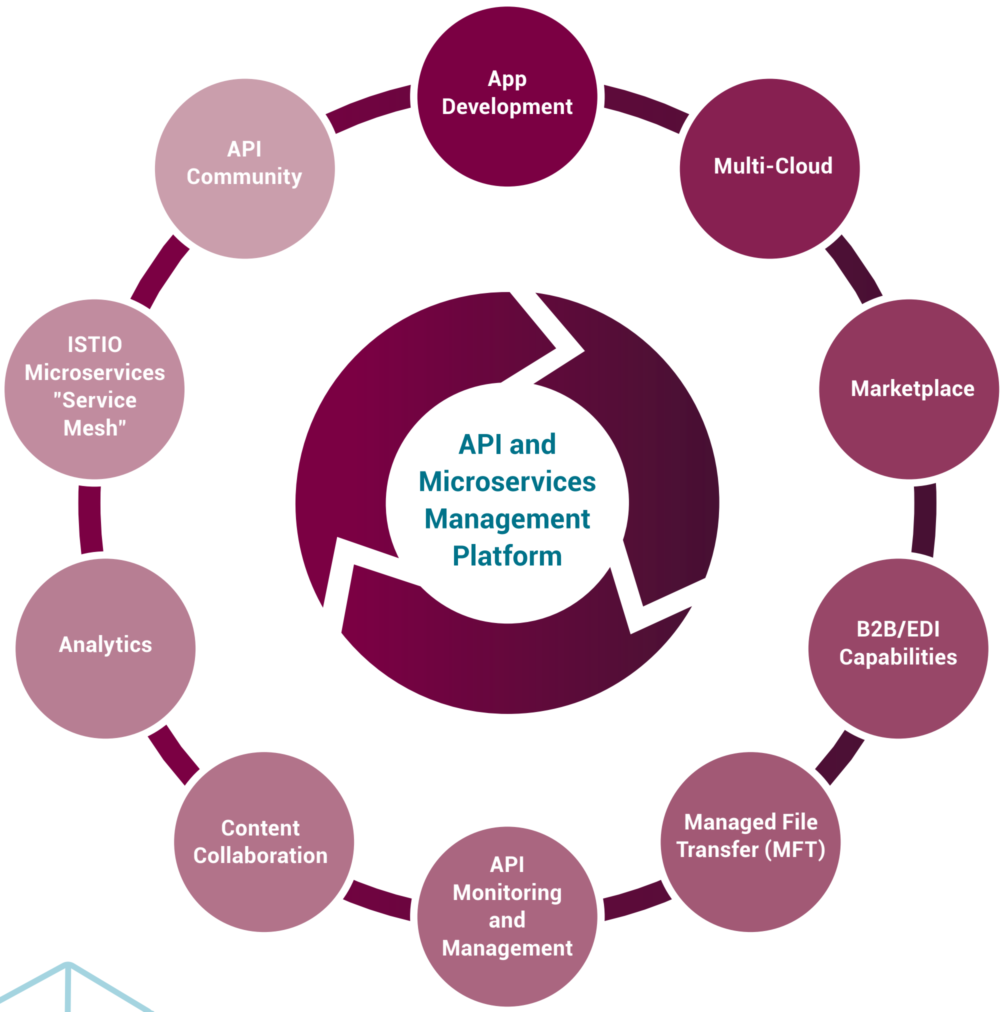
App Development
Multi-Cloud
Marketplace
B2B/EDI Capabilities
Managed File Transfer (MFT)
API Monitoring and Management
Content Collaboration
Analytics
ISTIO Microservices "Service Mesh"
API Community
API and Microservices Management Platform

Figure 2: *The new generation of API and microservices management platforms span a wide range of features, including application development, content collaboration, and analytics.*

CONCLUSION

Organizations that embraced SOAP web services and Services-oriented Architecture can make the move and realize the benefits of RESTful APIs. Doing so puts them on the path to transformative improvements in customer experience and digital transformation.

The use of APIs enables dynamic digital business at a scale first envisioned with the advent of Service-Oriented Architecture. The first step is to begin your transition to an API-based architecture. Modern API management plans and best practices help in the process. Key considerations include thinking like a startup and "failing fast" while starting on a small scale.

From there, the challenge is to expand access to APIs across the organization in a manageable way. Here, the API management platform can help a great deal, enabling the creation of developer portals and the consistent application of API lifecycle policies. The combination of best practices and an API management platform to provide consistency and control will establish the conditions for a successful API-driven customer experience transformation.

API's must be integrated, automated and pervasive. The management platform has to support the ISTIO service mesh, multi-cloud deployment, DevOps and CI/CD nativity. With these capabilities, APIs and Microservices can serve as the foundation for exciting new experiences for customers and partners.

What is an API-first strategy for digital business?

Discover how you can help your business bridge the digital divide.

WATCH VIDEO

Transform customer experience – and your business — with API management

How can API management make your business thrive?

LEARN MORE

Try AMPLIFY API Management online for free

SIGN UP TODAY

axway.com/en/products/api-management