openspan

Extending Legacy Applications to Consume Web Services

ACHIEVING SOA – NOW

Extending Legacy Applications to Consume Web Services

ACHIEVING SOA – NOW

INTRODUCTION

The process of developing a meaningful set of Web services for use in a Services-Oriented Architecture (SOA) is a long and involved one. It requires that an organization map out its core operations as a collection of discrete and largely independent services, build or otherwise create those services, and enable application developers to use these services in subsequent application development efforts. Even if these steps, and countless intermediate ones, are executed successfully, putting services-enabled applications into production use can take years.

In many cases one or more of these steps aren't successful, requiring enterprise architects or development teams to start over again or make significant changes to projects as they are in progress. Either way, it can add months or years to an already lengthy process.

Possibly the most difficult of these steps is in building the applications that can consume these services. Often the services are conceived of, designed, and constructed by the enterprise architecture team, while enterprise applications are built by application development teams. Often there are differing sets of requirements and goals between the two groups, and services built by enterprise architects may not be fully utilized in application development.

In many cases, the defined services may not even be Web services. If enterprise architects identify capabilities in legacy software, it may not be feasible to add Web front ends to those capabilities. Instead, enterprise architects may employ adapters, wrappers, or other technology that make those capabilities callable discretely from within the native legacy application.

Is there a way of consuming those services immediately, rather than waiting for them to be integrated into a brand new distributed application? Yes. Here's how.

It is possible to short-cut the long and arduous path between the development or other acquisition of Web services and the deployment and productive use of those services. Specifically, once services are designed and built, it is possible to employ them immediately with little development effort. By configuring both applications and services to be directly callable from existing applications, organizations can immediately take advantage of mission-critical services without a full application re-write development process.

The result is SOA - Now. The organization can take advantage of services it has developed before the entire SOA is built out. And it can deploy those services immediately, taking advantage of existing stable, trusted applications rather than being forced to go through a whole new application development lifecycle to get there. This paper describes how organizations can achieve the goals of their SOA strategy without the long and time-consuming process of deploying services in discrete applications to the enterprise.

This approach doesn't replace your existing SOA strategy. There is still a need to define and create Web services. And there are many circumstances where you would want to consume those services in the traditional way, through HTTP and SOAP. But there is often the need for business to move more quickly than traditional application development processes. This approach can accelerate the successful deployment of an SOA strategy by following the following four tenets:

- Focus on building the most valuable services
- Deploy immediately to the existing application set
- Focus your application development team on meeting the most important new requirements to support your business
- Decommission the legacy applications in a measured fashion over time

FROM WEB SERVICES TO SOA

One of the compelling advantages of SOA is the ability to flexibly respond to changes in business environment or opportunity. Services represent business logic that can be redeployed easily to new or modified applications in order to quickly meet new business needs.

The reality of implementation, of course, is a lot more complex. First, the discrete functionality in existing business logic can be difficult to identify. It requires a deep understanding of the underlying business processes, and the ability to identify the existing code associated with those processes. In some cases, it requires the writing of new code to automate manual process components, or to make automated processes more general for use outside of a specific problem domain.

Once logic is identified and partitioned, it has to be either packaged as a service or reverse-engineered and rewritten as a service. This typically requires a software development or integration effort (or both), adding Web services, front ends or adapters so that the business logic can be called as a separate entity.

As challenging as this sounds, given enough time, services can be identified and developed. It is primarily a technical problem with appropriate technology solutions.

But once business logic exists as a service, it must be consumed. Today, this requires an application development process, since the service is typically consumed as a part of a custom distributed application. There must be a need for a specific application that requires the service, and the application must be designed with the service in mind. The service or services have to be made available to the development team by the service owners, and scheduled to run on an available system. The development team has to be assured of the service's performance and scalability, and perform testing of the service within the context of the application.

And where the functionality of existing applications has to be re-written in order to take advantage of the new services, a significant integration and test effort has to be undertaken, with all the risk and cost that comes with it. The service, and other services, may well help the organization with agility in building applications that respond to changing business conditions, but it is a long and involved process to get to that point.

There is an easier and faster way.

The key to "SOA - Now" is the ability to consume Web services without first integrating them into applications using traditional programming techniques. This can be done using OpenSpan Studio, which enables the identification of programming interfaces available within existing applications, and the calling of those interfaces with specific data from other applications, services, or end users.

Further, SOA - Now using OpenSpan Studio also enables a more rapid identification of the code associated with existing business processes. By interrogating the running application for its interfaces, OpenSpan Studio makes it possible to quickly identify code that is associated with specific business logic.

USING OPENSPAN STUDIO

How does this work? Conceptually, OpenSpan Studio enables a services developer or integrator to interrogate running applications and identify interfaces that can be called through the OpenSpan Platform. In most cases, these are not documented interfaces, but rather defined internally for calls between different parts of the application. In many cases, they map to discrete pieces of business logic that can make up part or all of a Web service.

Note that this same approach also works for services defined by an enterprise architect or services designer. Even if the programming interface is defined and documented, as would be expected with a Web service, OpenSpan Studio can be used to expose properties and methods of the service to any new or existing application.

Once those interfaces are identified, developers can use them from within OpenSpan Studio to accept data from user input, applications, or other services through the use of a visual programming environment.

Developers can identify services within an existing application, find the interfaces to those services, and call them from any desktop-based application, web application or desktop front-end. This accelerates the delivery of business value from a SOA architecture by making it possible to implement the needed business solution on the timeline needed by the business.

SOA – NOW

The key to achieving SOA - Now is the combination of OpenSpan Studio for development and the OpenSpan Integrator runtime for execution. OpenSpan Studio provides a visual development environment that takes applications running in a Windows environment (including web applications, Java applets, 3270 host applications, and so on), and enables them to be connected in ways not envisioned by the original developers.

Take an application: any application. Let's say that the goal is to move data from that application to another application, and processed in a certain way in the second application. That second application is acting like a Web service, in that it accepts the data input, does a well-defined amount of processing, and returns a result.

It is possible to copy and paste that data, and perform the processing steps manually. Alternatively, you can extract the functional code from the application, wrap it, expose it as a Web service, and call it using a Web protocol. The first way requires human intervention and is likely to be slow and subject to error. The latter takes time and development resources.

Or it is possible to use OpenSpan Studio to write an OpenSpan "automation," which interrogates the applications for object properties, events, and methods, and uses them as a way to build a new application from the components of the others. An OpenSpan automation identifies interfaces in applications and automatically passes data into those applications using those interfaces.

BUSINESS EXAMPLES

Consuming Web services on the desktop may only be limited by the imagination of the workflow designers. This paper shows two hypothetical examples: consuming a sales tax rate Web service and extending a legacy Windows application to consume a web service.

Example 1: Consuming a Sales Tax Rate Web Service

A classic simple use case often used to describe how Web services and SOA can help business is that of sales tax calculations. Injecting variable rate calculations automatically to a workflow can be useful not only in sales or accounts receivable, but also in many businesses in the financial sector. Middle and back offices that work with settlements, credit extension, taxes, investment sales, and so on are prime candidates.

A regional business may have an application that tracks sales. There has been little need to be concerned about sales tax beyond the state in which it resides, because its customers have traditionally walked through the door and made purchases in just a few "brick and mortar" locations.

But such a business has likely found the need to maintain and grow by reaching out beyond its brick and mortar stores to the Internet. This means that the sales application now has to account for sales tax from different physical locations around the world. This has to occur quickly, or the business has to manually calculate and account for different sales taxes, a time-consuming process that in some ways defeats the purpose of the expansion to Internet sales.

For many such businesses, the only solution is to acquire and install a new sales application. However, using OpenSpan, the existing application can easily call a tax web service when calculating total cost for a customer order. Because tax laws change regularly around the world, the original sales application will likely not have this capability, or at least the updated data to effectively calculate tax. However, there are tax Web services available that have the necessary logic. The hard part is extending an existing legacy billing application to leverage the Web service. OpenSpan can do just that, by enabling the original sales application to consume the tax calculation Web service and integrate that data into the sales process.

In some cases, if the Web service has been developed internally, or is a commercial service, the interfaces will already be known. OpenSpan can still make use of those public interfaces to work with the traditional Web service architecture.

The steps to get to that point are simple and easily understandable by any application development professional. Using OpenSpan Studio, you identify an application running on the desktop (via the .EXE for a Windows application, the URL for a web application, and so on) and "interrogate" that application. The application launches, and you select a user interface object for inclusion in the OpenSpan environment. OpenSpan then exposes that object's properties, events, and methods, and you can use those in visually building a workflow of those components.

You visually build the automation by dragging connecting lines from one component to another. A connector, for example, can connect the output of one interface to the input of another. The end result looks like a workflow, and at runtime it executes and drives the application using the object components that you've selected.

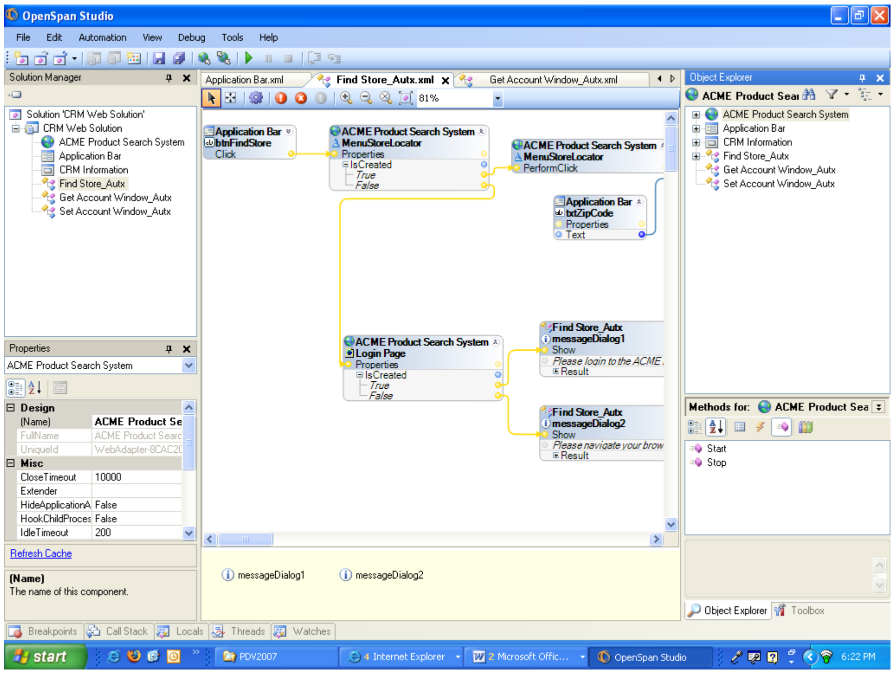


Figure 1. An OpenSpan automation is created by wiring together services based on their identified interfaces.

It is important to understand that you are not changing any of the applications, and you don't need source code for the application in order to build an automation. OpenSpan injects its interrogation into the running process, and leaves an interface for those objects you decide you want to access later. When you save an OpenSpan Studio project, the artifact is a "deployment package" which combines an XML description of the automation together with any dynamic content dll's that may be required at runtime. The deployment package is then executed by OpenSpan Integrator, the runtime environment that is installed on every desktop system where the automation is needed.

OpenSpan can interrogate and expose interfaces and objects from virtually any process. This means that just about any application, old and new, that runs on the corporate desktop environment including 3270 and 5250 host applications, Visual Basic, PowerBuilder and other WIN32 applications, Web applications (including Software as a Service), Java applications and applets can be analyzed and its functions turned into services. Those services can then be consumed by applications calling them from the desktop, or from direct input by the end user.

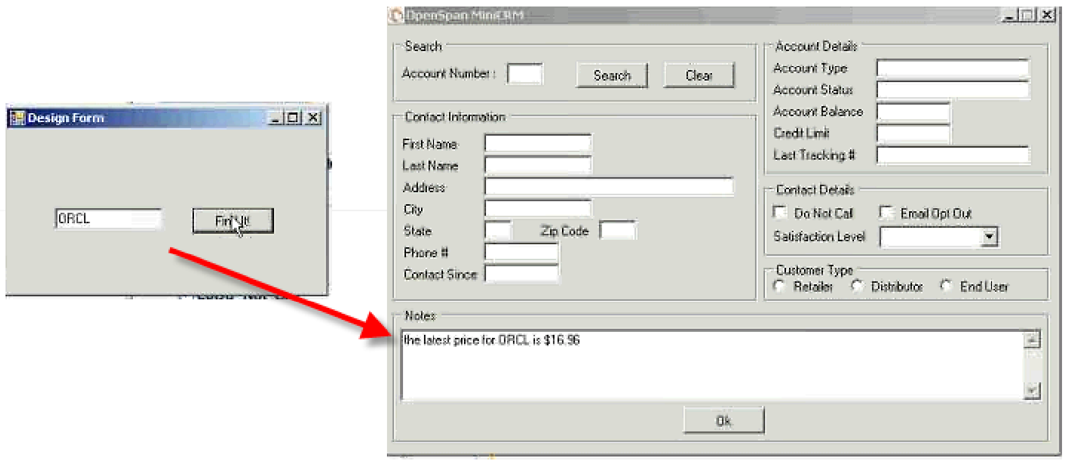
Example 2: Extending a Legacy Application to Consume a Web Service

This example calls a Web service in the traditional wrapper. The hypothetical case processes a stock symbol and produces a quote as of the query time, but the same design could be used in any case to take data from one application, process it, and carry the results to another existing application.

To make this happen, you:

1. Define the WSDL.
2. Build an automation.
3. Deploy and execute.

Figure 2. shows a simple test form (the source, with just a data entry field and an execution control button), and a Mini CRM window (destination). Mini CRM is a locally-written VB application running natively in the Windows environment, similar to many home-built applications currently deployed in many enterprises.

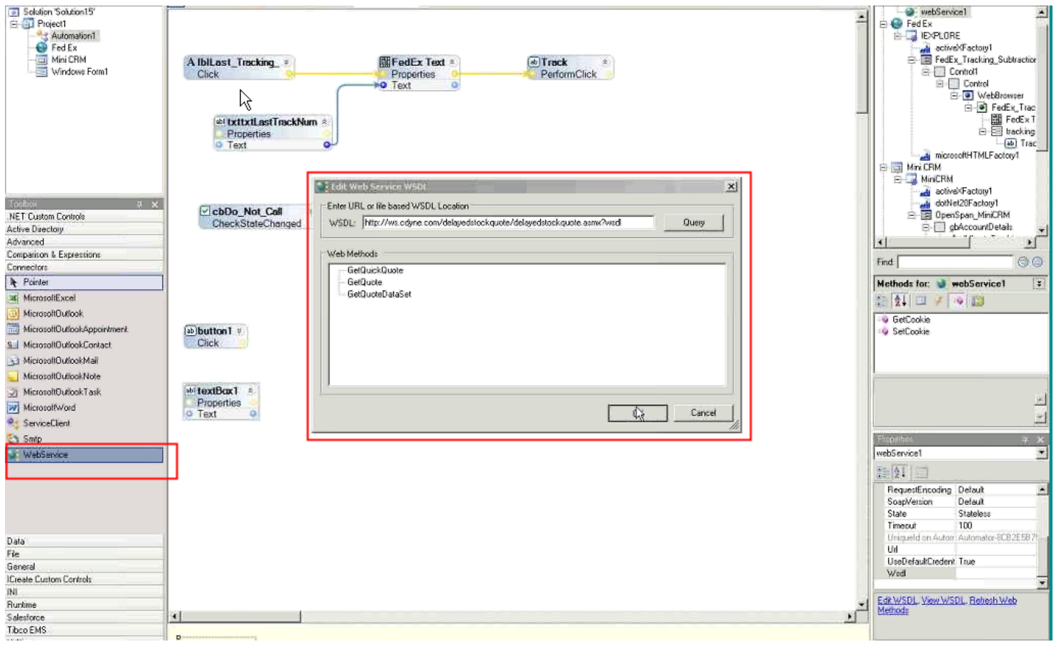


ORCL

the latest price for ORCL is $16.96

Figure 2. A form calls a Web service and displays the results in a different application window.

Define the WSDL. The first step in OpenSpan Studio is to invoke (Figure 3) a Web service to be consumed by Mini CRM. Drag a web service connector into the window. With a Web service, there's no visual component, so enter the WSDL in the active window, select Query to display the methods, and continue.



Edit Web Service WSDL

Enter URL or file based WSDL Location

WSDL: http://ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl

Web Methods
- GetQuickQuote
- GetQuote
- GetQuoteDataSet

Properties: webService1

RequestEncoding	Default
SoapVersion	Default
State	Stateless
Timeout	100
UniqueId on Autom	Automator-8CB2E5B7
Url	
UseDefaultCredent	True
Wsdl	

Figure 3. Defining the WSDL in the Edit Web Service WSDL window.

Build an automation. After interrogating the interfaces in OpenSpan Studio, build the automation (Figure 4). The required objects from the source form and the Mini CRM window are dragged from the Object Explorer to the work area. Connecting paths of execution and data is a matter of connecting the dots in the visual environment. Yellow indicates action, blue stands for data.

The automation uses a stock quote service's method, GetQuickQuote, that was exposed automatically by the WSDL Query process outlined above. Clicking the action button in the form calls the service and returns a result to Mini CRM. Note that you format the result using a standard string expression in the Studio window. (Remember that this form could be replaced by any application and the service call triggered by any event in that application.)

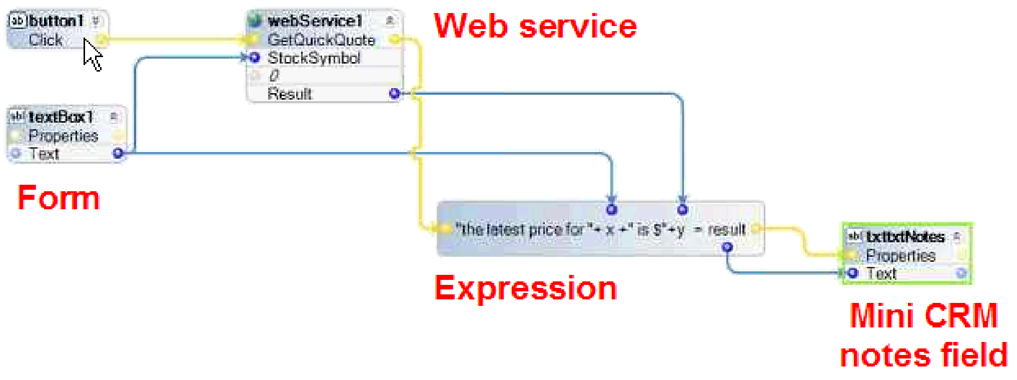


Figure 4. Connecting the new external service in OpenSpan Studio.

Deploy and Execute. When the solution is saved and deployed, the service executes whenever called, and delivers the formatted results to the target Mini CRM window.

OpenSpan is highly iterative, so you can reopen the solution, change or update, and redeploy quickly.

SUMMARY

With the OpenSpan platform, enterprises can more easily and quickly consume Web services, and can do so more quickly than through a traditional SOA initiative. The benefits of SOA are achieved more quickly, without giving anything up in the process. It accelerates the opportunity to derive value from a SOA initiative, allowing organizations to deliver those projects that directly impact the bottom line more rapidly.

Specifically, architects and developers can immediately apply new and existing services to building and deploying new functionality in existing applications. They can be wired together using OpenSpan Studio and deployed to the desktops requiring those services. These services are robust and high performing, and can be made available and are easily maintained and updated as required.

A process that can take years using conventional SOA strategies is reduced to months or even weeks of effort. SOA is typically a strategy that takes years to plan and achieve, with substantial technical decisions and challenges along the way. Focusing an organization's critical resources of talented architects and developers on only delivering new capabilities to support the business rather than on re-writing pre-existing code can give you the best of both worlds: all the benefits of a successful SOA strategy but delivered today with low risk and low cost. In other words, SOA – Now.

openspan

4501 NORTH POINT PARKWAY, SUITE 140
ALPHARETTA, GA 30022 USA
PHONE: **+1 678.527.5400** FAX: **+1 678.527.5401**