Composite Services

Executive Summary

Progressing towards an interoperable data and application model proceeds predictably as isolated islands of data and disconnected applications become unified through Virtuoso Web Services. The experienced CIO and IT department head is instinctively on guard for unhappy transitions to new systems infrastructure, almost to the point of expecting a scenario of notorious disruption, and occasionally, destruction.

In our case however, we have surmounted the two most vexing challenges of integration without resorting to a line of custom programming. Data Integration via the Virtuoso VDB is an administrative task, while API Function exposure was a matter of examining which application features required configuration via Virtuoso's Virtual Host and SOAP services system; thus far the transition has been remarkably free of the anticipated complications.

Composite services are examined in this monograph. Certain services may be grouped and invoked as packages, although we have created perfectly serviceable URIs that may, indeed, be invoked individually via the User Interface and application layer. It certainly makes sense to group duplicate functions across major LOB applications, when these operations would occur manually as a matter of course during data entry, at the very least.

In this particular case of merging two IT operations, we are destined to inherit applications from either side of the merger; as seen in the previous article's example of API exposure of the ERP and inventory systems, posting and updating of records will often need to be invoked in parallel. Many similar situations arise during a merger.

Creating composite services can be accomplished in many ways, using various methods. In this article's example, WSDL is used to bind multiple SOAP service calls into one composite service package. Be aware, however, that services may be invoked serially under server-side scripting control or as processes under Virtuoso's BPEL[1] manager (in our next article).

Binding multiple SOAP services via WSDL is easy to understand in the context of our use case, as there will be many instances of posting/updating multiple record sets to

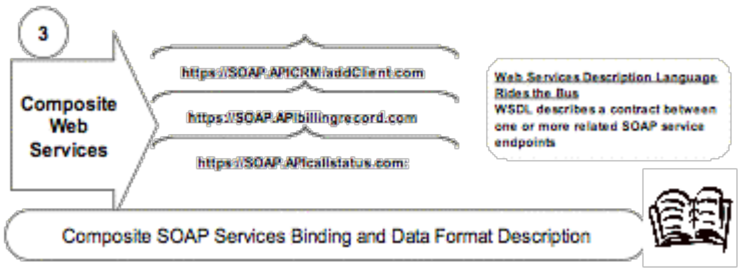
distributed systems - a common task outside of the web services world accomplished by other means.

Web Services Description Language - A language you really don't have to learn

If multiple SOAP-API functions are in your future, and you want to self document the calling and return parameters in order to automate consumption of your web services, then WSDL does the job. A proper web services platform should automate the generation of WSDL for your callers. If calling a WSDL service (as the consumer),the web services machinery should extract the parameters from the destination services and create the right conditions for implementing the services call. WSDL is very often compared to a 'contract of services', between web services callers and hosts; others may view WSDL as a cook-book for Web Services function consumers.

Back to the challenge at hand - invoking two related services to invoke add-client functionality. We have, in our previous exercise, exposed each API function in the ERP and Inventory system as SOAP services. We could use these services without regard to WSDL if we wish to a) rely on internal knowledge of the calling and return parameters (how it's been done for millennia), and b) if we invoke the URI functions by application logic residing at a higher level (also commonplace).

This ability to bind multiple services under one URI is known as 'Factoring Services'; the merger dynamic is littered with opportunities to group services [2]. In the case of posting data via a central service, it is perfectly legitimate to bind multiple Create, Update, Delete operations under


3
Composite Web Services
https://SOAP.APICRM/addClient.com
https://SOAP.APIbillingrecord.com
https://SOAP.APIcallstatus.com:
Web Services Description Language Rides the Bus
WSDL describes a contract between one or more related SOAP service endpoints
Composite SOAP Services Binding and Data Format Description

one WSDL definition dispatching multiple SOAP services endpoints.

In other cases, such as pending partner processes that involve long-running blocking wait-cycles for responses, we will use BEPL4WS, in order to create composite processes that orchestrate more complex actions between partners and internal processes.

For our example of collecting related Post Client Data API CRUDS, all you need is a few simple steps:

1. Choose your stored procedure(s)

The procedures that you want to expose can either be native Virtuoso stored procedures, or remote stored procedures that can be linked in using the Remote Procedures user interface.

2. Choose a virtual directory
   Because SOAP services need to be exposed and accessed via HTTP a Virtuoso virtual directory must be used. Either use the existing SOAP virtual directory or create a new one.
3. Publish procedures to virtual directory
   The procedures need to be granted read access to the SOAP user required. Use the Publish options on the virtual directory user interface.
4. Test the VSMX output
   Once procedures have been published as SOAP services they are automatically described by WSDL and testable using Virtuoso's VSMX feature.

Conclusion

The use of simple composite services to perform straightforward post and read operations is a Web Services power tool for merging dynamic application functions. Rather than writing application code to glue the partner systems together, we have exposed individual application functions via unique URIs, and created factored packages of logical functions via WSDL, and soon, BPEL.

Virtuoso Universal Server's powerful aggregation of Virtual Hosting and Web Services functions make fairly simple business of tying together the functions of our post-merger 'Add Client Record' function of two completely different applications.

One is always free to define User Defined Types that call individual SOAP end-points, and/or write application code that loops through the native stored procedures, however, the elegance of Virtuoso's exposure of service invocation URI's and composite binding into WSDL description files is a far more elegant and maintainable way of building complex systems via composition of services.

Learn More

- **Virtuoso Documentation**
- **Virtuoso Tutorials**

[1] Business Process Execution Language - a W3C XML standard for creating and ordering web services processes from the top level, as opposed to embedding process handling in application code.

[2] See as explained in 'A Super platform for Mergers', link here*