

Process Orchestration

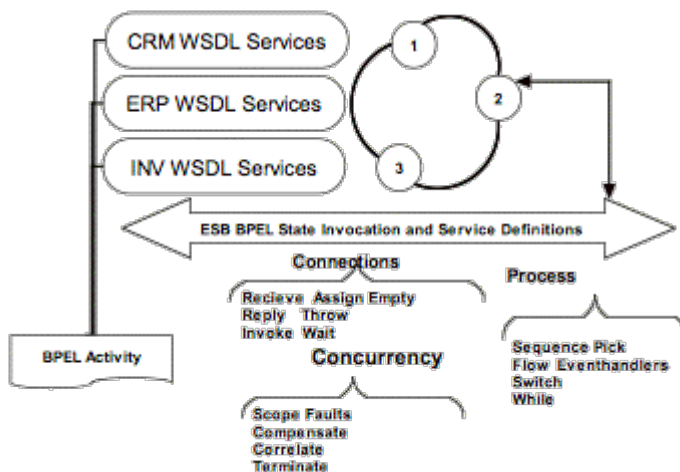
Executive Summary

The arc following our steps from isolated systems, through data integration, function exposure, and composite services, has been quite a journey. Stepping back and examining our handiwork, we have created a supremely flexible web services data infrastructure, without disrupting out foundational applications and operations in the least. The last item in this series of articles will cover process management using the latest maturing standard for Web Services and SOA architectures, Business Process Execution Language for Web Services (BPEL4WS or BPEL).

Virtuoso Universal Server incorporates a BPEL engine that allows one to create a composite process layer one level of abstraction above the WSDL executable end-points explained in the previous articles. BPEL is a medium for creating composite WSDL processes with the added benefit of process coordination internally or with external partners.

BPEL is an evolving yet rapidly maturing standard; we naturally expect our tools to be as transparent as possible, hiding as much complexity as can be expected.

In this article, we will examine the strategic use of BPEL as an integration process enabling tool. BPEL4WS - Business Process Execution Language for Web Services



Drawing a page from current and past practices, we can see that the composite services we have created from individual executable application end-points are ready for invocation - yet where will these services be coordinated? Typical programs of yore would have master event loops pending on our partner processes. More contemporary, pre-web services applications have had applications call service container classes that handled messages in queues, etc.

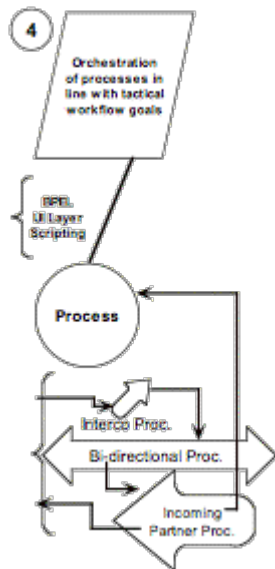
Today, we have standardized Web Services built atop XML, SOAP, WSDL - in that order, and now BPEL. BPEL extends the WS model by allowing composition of your WSDL services into processes. BPEL is a W3C XML standardized language that directs business processes from the top down. Whereas one used to code the event loops or custom messaging scripts to pend or spawn processes, we now create BPEL Partner Links, Activities, and Ports, which call and combine our executable URI services. It's a very different paradigm, and one that seems to make more sense, as standardized XML language modules (BPEL) used to create and service processes can be archived and exchanged for commonly used services. BPEL allows a natural division of process orchestration separated from the application logic called in your WSDL executable URI's.

Bits of modular BPEL code are becoming increasingly available for all types macro processes, such as loan approval processes. This evolution mirrors the modular code library revolution, but from a top down and WS standards-based POV. It's an improvement over proprietary MOM [\[1\]](#) systems, DBMS EAI, and vendor specific processes orientation, such as MS BizTalk (Which is, as of the writing of this article, catering to BPEL, as is the Oracle J2EE camp!).

BPEL In Process Composition

As a way to manage pending processes internally or between new partners or internal systems integrations, BPEL fits into a web services integration strategy quite nicely. BPEL deals only with business processes: control flow (branch, loop, parallel), asynchronous event handling, long-running nested units of work, faults, and recovery methods. Do not confuse BPEL with work-flow - there is no concept of users or people. Rather, think of BPEL processes as collections of web services enabled URIs, expressed as WSDL, and subjected to definitions of process, procedures, and partner relationships.

The meta model of BPEL is expressed as two types of partner process definitions: 1) Abstract Processes, and 2) Executable Processes.



Abstract Processes are BPEL services that expose only the public aspect of a business process, without regard to how partners might implement the execution of a service. An Executable Process exposes the complete state of the details involved in determining the sequence of interactions of the partners. Executable processes are likely found in integration scenarios where the partners (even internal departments), must have certain system and session dependent data shared, for expediency. Fully abstracting a process, in any system or programming model, may be difficult in certain circumstances.

There is no sharp division for the need of deciding between the Abstract and Executable, as processes need not be fully defined in private implementations. BPEL is merely a neutral web service class for business processes using XML data.

Merger dependent processes will execute against partner services consistently, no matter where you 'draw the line' in your infrastructure or code. In other words, one can compose and expose web services, couple these services to BPEL partner links and ports (services collections), and the rest is anyone's game - one is not required to fully define the process in BPEL.

Conclusion

The beauty of BPEL4WS is the incremental nature of how it fits the web services model. Harking back to the terminating paragraph of the last section, we can indeed envision certain simple processes defined for the dispatching of simple, synchronous processes that return ready results in a single session. As our maturity inevitably develops across the entire web services spectrum, and as we develop more exposed code and SQL procedures, our opportunity for deploying BPEL will increase and we can match up more complex business processes for receiving asynchronous results.

But the most exciting news is that BPEL4WS complexity will likely fade as more design-time tools hide the details from our tender hearts; UML, IDE's, and new front-ends will likely make the nitty-gritty of BPEL less irksome. Starting gently with the Virtuoso Forms based BPEL tool, and persevering, will likely net a GUI based design

Learn More

- [Virtuoso Documentation](#)
- [Virtuoso Tutorials](#)

[1] Messaging Oriented Middleware