SearchSOA.com E-Guide

Data Integration

The SearchSOA.com E-Guide: Data Integration provides an expert overview of the role of data integration in SOA development. It discusses the prominent function of data services platforms (DSPs) in an SOA initiative and offers insight into the seven levels of loose coupling that architects should consider when developing this infrastructure. This E-Guide details the importance of a Data Services layer built upon an enterprise data integration platform for SOA success.

SearchSOA.com E-Guide

Data Integration

Table of Contents:

Data Service Platforms for SOA Emerging

By Rich Seeley

This may be the year that data services platforms (DSPs) emerge from the shadows of the enterprise service bus (ESB) to take a more prominent role in service-oriented architecture (SOA), according to a new report by Burton Group Inc.

DSP technology first came on the scene in 2005, but at that point SOA products were focused on infrastructure, especially ESBs, according to "Data Services Platforms: Searching for Their Place in the Market," a 53-page report published this month by Burton.

"Now, at the beginning of 2008, the future looks a little brighter for the DSP market," write the report's authors, Anne Thomas Manes, research director for Burton, and Stuart Selip, vice president of consulting services for the analyst firm. "Market awareness has improved. DSP vendors are finding that many prospective clients already understand the value of data services, and they are actively looking for a solution."

Asked who in the IT department needs to be focusing on DSP technology and products, Manes said it crosses boundaries through the SOA development lifecycle.

In the beginning, the selection of DSP products will probably be the responsibility of an architect, she said. Depending on the company, that specific title might be enterprise architect, data architect, service architect or integration architect, she said.

"At development time, it might fall to the data integration team or the SOA team or, in situations where you don't really have an SOA team, the application integration team," she said. "One of the primary reasons that the EII [Enterprise Information Integration] vendors switched their marketing message to services was to attract attention from the SOA team because they weren't making any headway with the data integration team."

Once the cycle moves to runtime, she said the responsibility for DSP moves to the operations organization, which is responsible for application servers and similar infrastructure.

For all those interested in the technology, the Burton report contains an explanation of what data services and DSPs are and why they are important to SOA:

- A data service is a type of service that provides access to data—any type of data. Essentially, a data service can enable access to any data source that can be accessed programmatically. It combines concepts from data integration (DI), application integration, and service oriented architecture (SOA) domains to enable access to live data regardless of its physical location or format. A data service provides an abstraction layer that hides the physical characteristics of the data source as well as the mechanisms used to access it. From a service consumer's perspective, the service is the data source, providing a simple, convenient, logical data model that can be accessed and queried using various query mechanisms."

- "The demand for data services tooling has spawned several data services platform (DSP) offerings. These products provide value from both a tactical and strategic perspective. DSPs support rapid development of

> data services that enable ad hoc access to and integration of a variety of data sources—all the while abstracting the underlying access details."

Helpful as all this sounds, Manes and Selip caution that DSPs are not a silver bullet and in fact require SOA developers to work to make data services work.

"A DSP is not a panacea," the authors explain. "It is not an out-of-the-box, all-encompassing data access layer for all enterprise data. Data services must be built one at a time, and each service provides access to a specific set of data."

Handy as the DSP may be, it does not automate the creation of data services. Nor should a DSP be thought of as a replacement for a data warehouse or traditional data integration system, but as a complement to them, they write.

After cautioning against great expectations, the authors suggest that SOA architects and developers focus on initially using DSPs on targeted projects that have the potential for a noteworthy return on investment (ROI). Dashboards, customized report generators and data integration for a specific application are among the high-value projects they recommend. Data services mashups are another area where DSPs can proved their worth.

The authors also suggest downloading free trial versions of various vendors DSP products and trying them out before buying.

The Burton report offers a detailed analysis of the DSP vendor landscape. Two vendors with the greatest "mind share" are BEA Systems Inc. (which is currently being acquired by Oracle Corp.) and Progress Software Inc. AquaLogic DSP is the BEA product, but the report notes that Oracle has not yet offered guidance on how it will fit into its SOA and data management product line. Progress offers DataXtend Semantic Integrator based on technology it acquired from Pantero, which had been a pure play DSP vendor.

Red Hat Inc., is also listed as a player with its acquisition of MetaMatrix in 2007.

MetaMatrix and Pantero will not be the only pure play DSP vendors to be gobbled up in the dog-eat-other-dog's-homework world of SOA, Manes and Selip predict. Three big dogs in SOA infrastructure, IBM, Microsoft, and Oracle Corp., may continue to hunt for pure play vendors.

Among the pure plays, the authors list:

- Composite Software Inc., a DSP and enterprise information integration (EII) vendor
- Denodo Technologies Inc, a data mashup vendor
- Ipedo Inc., an XML data management vendor
- WSO2 Inc, open source SOA infrastructure vendor
- XAware Open Source Integration Community, data integration vendor

About the author: *Rich Seeley is an experienced technology writer and a key contributor to SearchSOA.com.*

The Seven Levels of Loose Coupling

By Ronald Schmelzer

When ZapThink talks about service-oriented architecture (SOA), we generally try to avoid semantic arguments about what is and what isn't SOA, but rather try to focus on specific characteristics that identify service-oriented systems and the benefits they provide organizations that adopt the architectural approach. In particular, we like to focus on three core tenets: loose coupling of service consumers and providers, composability at a variety of levels of granularity, and event-driven, asynchronous styles of interaction that allow for a wide range of usage scenarios.

While these characteristics might seem distinct, they are in fact related to each other. The ability to compose arbitrary services in environments of long-running transactions requires a certain measure of loose coupling. However, architects seem to grapple with the term "loose coupling". Some erroneously think that systems are either loosely coupled or tightly coupled as if the determination of coupling is a binary evaluation. The reality is that the concept of coupling, like that of granularity, is a relative term. Something is more loosely coupled than something else if it allows for a greater degree of variability and freedom for that particular aspect. Simply put, coupling is a measure of dependency and commonality required by a service consumer and provider to be able to achieve a result.

In that light, loose coupling is a spectrum. A system can be tightly coupled in one aspect while being loosely coupled in another. Many architects say that they want complete decoupling of systems so that any variation in the system can be handled without having to re-implement any of the service consumers or providers. As we've discussed in past ZapFlashes, complete decoupling is incredibly difficult and expensive, if not impossible. So, what architects should be aiming for is achieving the right level of loose coupling to facilitate business agility without imposing huge costs.

As such, what ZapThink has seen is that there are really seven levels, or perhaps aspects, of loose coupling that architects should consider in their SOA initiatives. The more degrees and levels of loose coupling they add to their SOA efforts, the greater those systems can deal with change.

Loose coupling the implementation

One of the most fundamental promises of SOA is that service consumers are blind to the implementation technology used by the service providers, and vice-versa. Indeed, the whole idea of the service abstraction is that technology implementations are hidden. This most basic of SOA characteristics is realized using standards-based, interoperable specifications or protocols that provide a contract-based interaction between service consumers and providers. Web services certainly fits the bill as does REST and other styles of service implementation. Indeed, it's not the choice of interoperable specification (whether XML based or not) that enables this level of loose coupling, but rather the service contract. A properly written service contract will allow service consumers to bind to a wide range of service provider implementations without have to know at all whether the service implementation is done in Java, .NET, PHP, C++, or Basic. All SOA implementations must be at least loosely coupled at the implementation level, but clearly this is not enough to guarantee the sort of complete loose coupling so desired by the business.

Loose coupling the service contract

XML, Web services, REST, and other specifications hide the implementation of the service such that when the service implementation changes, the service consumer does not need to be the wiser about it. However, what happens when the service contract itself changes? A simple change to acceptable inputs or functional behavior of the system can have profound impact on service consumers. As such, architects that want to progress their SOA initiatives beyond simply Web service integration need to address service contract change management in such a way that contract changes don't cause service consumer breakage.

This might seem to be a difficult proposition, but there are plenty of best practices, architectural approaches, and technologies to facilitate this level of loose coupling. First, a proper service contract that is interface-neutral will alleviate much of the problem of service contract change management. The Service Description Framework (SDF) shared by ZapThink in our Licensed ZapThink Architect (LZA) boot camps is one of those technology-neutral service contract templates. In those approaches, new service contracts don't simply replace old ones. Old contracts are never deprecated unless a transition path is provided for service consumers. This can be as simple as a transformation or as complex as a service-oriented process that is triggered when service consumers request old (and deprecated) versions of the service contract. As such, contract change must be handled as a matter of policy. Who gets what version of a service should be controlled via metadata, not programmatically or via configuration of tightly-coupled infrastructure.

Active management and service intermediaries further simplify the service contract change management issue by providing exception management and handling of service contract differences. And of course, SOA Governance plays a huge role here in helping to manage the sort of change and versioning issues that arise. But probably the best practice in this arena is the use of late-binding approaches that leverage run-time registries and contract-based binding to abstract the binding specifics of service consumers and keep service contract changes from propagating. The topic of late binding is itself a deep conversation that requires rethinking the way that service consumers bind to service providers, and one too long for this ZapFlash.

Loose coupling the service policy

So, putting into play service contracts that abstract implementations and late-binding, intermediary-enabled, registry-based systems that allow for service contract change without breakage enables a great degree of variability in the system, but we're still far from done. Indeed, even if the service contract stays stable, a small change to a service policy can have tremendous repercussions, as discussed in our Butterfly Effect ZapFlash.

Companies looking to address service variability should handle changes to policy the same way they handle changes to service contract: late-binding, service intermediary-enabled, registry-based, and governed. A policy is a form of metadata, as are contracts, and in fact, the only difference between a service policy and a contract is that a policy can apply to any number of services. Because policies control many aspects of the non-functional parts of a service, architects need to include in their architectural plans, methods and practices for dealing with service policy versioning and deprecation as well as policy versioning that leverages the technologies and practices established for service contracts. Companies need to test policies just as they test service implementations, and manage

policies as rigorously as they manage services. Doing so not only makes the system as a whole more reliable, but enables one more level of loose coupling.

Loose coupling the process

Of course, loosely coupling the service consumer from a service provider only provides agility if the services aren't composed together. Once you have a business process that composes a bunch of services together, we have another area of potential tight coupling. What happens when the process changes? Ideally, a service consumer should not have to know at all when a process is reconfigured. Fortunately, achieving that level of loose coupling is fairly straightforward.

The movement in the SOA and BPM markets has been towards separating the process definition layer from the underlying implementation. By defining processes in metadata and exposing those processes using service contracts (the recursive vision of process compose of service and exposed as service) abstracts the implementation of the process from the service consumer. In fact, in such a scenario, a business process is really a form of service implementation. And just as service contracts provide loose coupling of service implementations, they provide loose coupling of service-oriented processes.

Loose coupling the data schema

If we've enabled all the loose coupling levels defined above, companies should be able to make arbitrary changes to their service implementations, contracts, policies, and processes without breaking a thing (or a sweat). Yet, that's not sufficient to handle the changing needs of the business. What happens when the underlying data schema changes? If a service consumer and provider need to have a common understanding about that in order to have a conversation, we have tight coupling as defined above. Therefore, organizations need to further their loose coupling goals by enabling dynamic and heterogeneous change to the data schema shared between service consumers and providers.

To some, this might seem to be a very complicated task. Computers aren't people, after all, and so they can't process any changes to the format or definition of data. Yet, not all hope is lost here. First, services aren't really interoperable unless they can understand and process data in common. This means that while there might be semantic dependencies between them (which we will address shortly), there does not need to be structural data dependencies between them. Schemas are key to service data interoperability. However, what happens when Schemas change? One key approach is to address information and schema management in the same way you approach service metadata management. Data schema can be treated as a form of metadata and the use of exception management, transformations, service intermediaries, and Data Services are all key to enabling loose coupling of data structure. Implementing a Data Services layer introduces loose coupling in a way that provides a separation of concerns, so that underlying data infrastructure changes are insulated from the business services above.

Furthermore, companies need to align the efforts of those maintaining the data schema with those maintaining the service metadata. Why is it that the folks who maintain the schema are not part of the architectural team? Schema are no different than service contracts. They represent a form of metadata that encodes business requirements. As such, we frequently admonish companies to bring their data and information architect teams into their enterprise

architect fold. The simple act of this alignment and the establishment of metadata-enabled change management will allow for loose coupling of data schema and structure.

Loose coupling the infrastructure

Before we try to slay the last dragon of loose coupling, it's important to note that all these levels of loose coupling doesn't matter a whit if it all depends on a single vendor's implementation. Too many times we interact with architects who claim that their systems are loosely coupled, but if they move their implementation from one Enterprise Service Bus (ESB) or service infrastructure to another then all hell will break loose. How can they truly say their implementations are loosely coupled with such a huge dependency in mind? Enterprise architects worth a grain of salt will demand that their service implementations be infrastructure neutral. That means that at any time, the company can change their infrastructure without having to rebuild all the service consumers and providers.

Many vendors promise this sort of interchangeability, but few deliver. In fact, the industry as a whole seems to be moving towards single-vendor SOA platforms that will keep many SOA initiatives tightly coupled at this layer. This ZapFlash serves as warning to firms that want to achieve high degrees of loose coupling: keep your implementation infrastructure neutral and you will be successful. Otherwise, your SOA initiatives will only enable partial business agility.

Loose coupling at the semantic layer

The final level of loose coupling is the most challenging of all. Even if a company is enabled to make all the changes it wants to service implementation, contract, policy, process, infrastructure, and data structure, problems still arise whenever there is change to semantics. As we detailed in our Semantic Integration: Loosely Coupling the Meaning of Data ZapFlash, if we impose the data structures of Service providers on service requesters, the result is every bit as tightly coupled as previous architectural approaches. In order to provide the promise of seamless data integration, we must transcend simply loosely coupling the application interface and in addition provide loose coupling at the semantic level.

As we detailed those ZapFlashes, in order to achieve the sort of semantic data integration we are seeking, we must implement dynamic service definitions. In essence, the definition of the service interface must change based on the context of the service requester. As a result, a service can change its contract between invocations. For example, the fact that a service provider requires first names to be no longer than 40 characters should not require the requester to know that fact. The contracted service interface is supposed to provide that isolation. Service interfaces must therefore become much smarter. Instead of having to know ahead of time what specific data requirements are needed by a service, the service requester should be able to dynamically discover a service interface that can not only provide the needed functionality, but also understand the information payload.

The ZapThink take

SOA as a whole is an exercise in loosening the coupling of heterogeneous systems that have to cooperate to meet the needs of the business. As such, as companies seek to mature their SOA initiatives, they should be seeking to iteratively loosen the coupling of their systems at the various levels described above. The most amazing part of this

loose coupling exercise is that the greater the degree of variability that can be tolerated by the system, the greater agility that you've enabled for the business. Even more so, if companies can manage to address all seven layers of loose coupling while maintaining a flat cost of architecture, then we've reached the desired state of the IT-Business relationship: IT can implement every requirement and desired outcome expressed by the business without imposing any additional time or cost impedance.

To be specific, in a system that is enabled with all seven layers of loose coupling, changes to service implementations, business processes, service contracts, service policies, runtime infrastructure, data schema, and semantics won't break the system. Can you think of any requirement by business not addressed in those levels of loose coupling? This vision of agility is precisely the aim of SOA, and it clearly requires much broader understanding of SOA than simply loose coupling of the implementation, which is the only capability provided by Web services.

About the author: *Ronald Schmelzer, senior analyst and founder of ZapThink, is a well-known expert in the field of Service-Oriented Architecture (SOA), Web Services, and XML-based standards. Ron has been featured in and has written for periodicals, and has spoken at numerous industry conferences and in front of some of the largest businesses in the world.*

Enterprise Data Integration: A Critical Piece in a Service-Oriented Architecture

By Ivan Chong and Ashutosh Kulkarni

A large company found itself handicapped by an ornery snarl of siloed applications that compromised its agility, performance and profitability. Its IT department was constantly behind schedule and over budget in hand-coding point-to-point connectivity among supply chain, financials, CRM, and other packaged and custom-built legacy applications.

The solution: Integrating critical business processes and applications by adopting a service-oriented architecture, or SOA. Internal IT personnel and consultants engineered a loosely coupled infrastructure, with reusable services based on XML and standard Web services protocols such as SOAP and WSDL. Once the system went live, the CFO ran a routine query through his dashboard. The answer came back:

You forgot the data.

It's a playful fiction, of course, but it illustrates the perils of a service-oriented architecture that focuses only on the business process interactions and application interfaces, and neglects the devilish details of data-level incompatibility among the disparate IT systems participating in those processes, including varying formats, semantics and hierarchies.

Our hypothetical company based its SOA on a Web services-based enterprise application integration (EAI) engine. The technology worked flawlessly in enabling high-level application integration and orchestrating business processes —but it was not designed to deal with the complexities of heterogeneous, inconsistent, dirty data that lies fragmented across the enterprise.

The result: Costly and time-consuming hand-coding to resolve these data inconsistencies in the SOA implementation, thus violating the very promise of reusability and interoperability that is driving the movement towards SOA. The missing ingredient in this company's SOA was a Data Services layer built upon an enterprise data integration platform.

The SOA opportunity

The buzz around SOA has been fast and furious. It's no wonder. Organizations recognize an opportunity to slash the cost of application and middleware development and accelerate time to market by "loosely coupling" siloed applications using open standards such as Extensible Markup Language (XML), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL).

The widespread adoption of these standards by IT organizations and vendors alike paves the way to expose applications as component-based services for delivery over the Web. By abstracting the underlying business logic, SOA

enables services to be wrapped, re-used, and orchestrated to give both IT and business far greater responsiveness, flexibility, and speed of execution.

Many early SOA-based implementations have been built on EAI, and J2EE- and .NET-based middleware, including message brokers, application servers, and enterprise service buses. But increasingly, data integration has become a primary objective. Some 76 percent of AMR Research respondents using or planning to use an SOA named process or data integration as the leading initiative, according to the August 2005 AMR Research report, "Service-Oriented Architecture: Survey Findings on Deployment and Plans for the Future." The findings reflect a growing awareness that a data integration platform can—and should—enrich an SOA with sophisticated data services beyond the scope of application integration-centric technologies.

In other words, to realize the full potential of SOA, including loose coupling and reusability, it's critical that client application access business-relevant data wherever is resides, in whatever form it required, in a consistent and accurate manner.

Ready for prime time: Service-Oriented data integration

Data integration technologies are ready to help SOA become a transformative force for IT. Over the past several years, data integration technology has been enhanced with built-in support for XML transformations, Web services protocols, JDBC connectivity, and Java Message Service (JMS) connectivity. Advanced data integration platforms also feature metadata capabilities driving the core of their development and run-time infrastructure. This metadata provides an abstraction of the business logic from the technical implementation, and enables them to deliver advanced data integration functionality over a data services layer to the myriad components in the SOA.

For too many years, data integration initiatives, undertaken without the foundation of a data services layer, have resulted in a further proliferation of the siloed systems they were meant to integrate. For instance, a retailer might have deployed an extraction, transformation, and loading (ETL) tool to synchronize point-of-sale data from retail outlets into an SAP financials application. A second instance of the tool might serve to move SAP financials information into a DB2 data warehouse for analysis. And a third instance might work on the front end of the value chain to feed product procurement data to an operational data store.

So while the retailer will have achieved data integration among targeted applications, it's still several steps removed from realizing a fluid, end-to-end data ecosystem. SOA removes these barriers of siloed development.

In a modular SOA, a data integration platform serves as another component-based service. Its functionality can be packaged and reused across multiple projects to reduce development and deployment costs. It can help an organization leverage data assets currently locked in mainframe, packaged, and homegrown systems through open standards. It can eliminate the need to hand-code data integration connectivity, and enables businesses to realize rapid time to value.

That's what SOA offers data integration technology. Now let's look at the flip side - what data integration does for SOA.

Data components and services in an SOA

The most advanced SOA deployments will take advantage of both EAI and data integration technologies. SOA provides an ideal framework for these two technologies to complement one another, with EAI managing transactions and processes among applications, and the data integration platform performing the atomic-level data processing that is generally beyond the scope of EAI systems.

In fact, a common use case is where a company deploys an EAI bus and a data integration platform in a SOA to support master data management initiatives, such as customer data integration. The EAI bus drives business processes and checks customer records in the master data repository. The data integration platform creates the master data repository, and populates back-end ERP systems with updated customer information transformed to the appropriate format and semantic definition. In strategizing options and objectives for an SOA, organizations should assess and understand the functional distinctions between the two technology sets. Let's take a look at three functional components that are the exclusive province of data integration technology - universal data access, a metadata repository and services, and a data integration engine.

Universal data access: Scope of data

Data integration extends the reach of the SOA and its constituent applications into virtually any data source. Prebuilt connectivity and visual mapping environment provides IT architects and developers with a mechanism to tap into information from a variety of sources, including packaged and homegrown applications such as SAP, mainframe and midrange systems such as IMS and VSAM, relational databases such as Oracle and Sybase, and unstructured and semi-structured data.

Organizations can use data integration to reach into multiple systems to fetch data, cleanse and transform it into the appropriate formats and semantic definitions, and propagate it across multiple distributed systems. Its service may be invoked by, for instance, an online customer order application to trigger event-driven, read/write data updates across financials, manufacturing, and distribution in near-real time.

Metadata repository and services: Meaning of data

A metadata repository provides the SOA with an underlying foundation to understand the lineage of data, the ripple effects of changes, and data-related deficiencies in the architecture. The repository provides a data interaction framework to store and manage data models, transformations, workflows, and dependencies- metadata describes the data logic and its meaning. Through metadata services, data integration technology provides a means to reconcile data semantics across disparate systems, improve reporting, auditing, and data governance and enable reuse to streamline development and accelerate deployment.

Metadata is also key in equipping organizations with an auditable record of data lineage covering all data resources, providing an important tool in meeting the compliance requirements of Sarbanes-Oxley and other regulations.

Data integration engine: Value of data

At the core of data integration is an engine that provides organizations a host of options for moving, integrating,

and delivering data among various consumers in a SOA. Its flexibility is important in letting IT professionals architect a system optimized for "right time" data delivery, including high-volume batch data movement, near-real time capture and movement and changed data capture - only data updated since last service invocation.

Data integration also offers functionality to help "future-proof" an SOA against rising data volumes, meet the requirements for reduced data latency, and demands for toughened security and privacy. For example, data integration supports partitioning to optimize parallel processing on multi-CPU hardware, deployment on multi-node server grids for distributed workflow execution and fault tolerance, failover, and fortified security through authentication, authorization, and encryption.

At your service: Data dividends in an SOA

So those are the core data integration components in an SOA. Now let's examine the services and benefits that they deliver for an SOA—data profiling, cleansing, transformation, movement, and auditing.

Data profiling: Data profiling is the process of assessing and understanding the content, quality, and structure of enterprise data. It is an essential step in reconciling semantic differences in common business vocabulary such as customer, address, and product that varies among applications, and which, if unaddressed, results in contradictory information across the enterprise.

Data cleansing: Once data is profiled, a data integration platform can execute data cleansing functions to ensure the validity and consistency of information. It standardizes name, address, and other values, and resolves missing data fields, parses data elements, and corrects poorly formatted or conflicting data.

Data transformation: Data transformation services enable data to be transformed from one form to another to allow reconciliation between data elements residing in different information sources. The transformation services leverage pre-built and customized mappings that take into account complex data hierarchies and relationships.

Data movement: Data integration offers flexible mechanisms for "right-time" data delivery in an SOA, including high-volume bulk data movement, near-real time capabilities, data federation, and changed data capture that handles only information that has been updated to accelerate load times and minimize operational impact.

Data auditing: Data integration provides in-depth lineage of data - when it was changed, how, by whom, and across which applications—to enable auditing, reporting, and analysis essential to meeting the demands of legislated regulations and internal/external auditors.

The data services provided by the data integration platform can be accessed by other components in the SOA via Web Services protocols such as SOAP, messaging systems such as MQSeries or JMS, and programmatic approaches such as JDBC and ODBC.

Where do we go from here?

SOA may still be in its early phases, but the time is right to take a hard look at data-related business objectives

and IT resources in your service-oriented architecture blueprint. One key to success is an iterative approach that focuses first on targeted projects with quantifiable business value that are relatively easy to implement. SOA, after all, is a matter of architecture, and no organization is going to rearchitect its systems overnight.

By implementing a data integration platform at the ground level, you can ready your IT systems to fully leverage that most valuable of business assets—data—without re-engineering, hand-coding, and having to worry about data quality problems down the line. Plus, you'll never worry about receiving another response that says, "You forgot the data."

About the authors: *Ivan Chong is currently General Manager for Informatica's Data Quality Business Unit. He has also headed up Informatica's Product Marketing as well as worked in the Corporate Strategy group doing business development. For much of his career at Informatica, Ivan served as Vice President of Product Management.*

Ashutosh Kulkarni is Sr. Director of product management and marketing at Informatica, where he leads the teams responsible for Informatica's core data integration products, including PowerCenter, their flagship data integration product, and PowerExchange, their family of data access products.

Resources from Oracle

ORACLE

« **White Paper: Oracle WebLogic Server - A Solid Foundation for SOA**

« **White Paper: Oracle IT Modernization Series Modernization: The Path to SOA**

« **White Paper: Oracle WebLogic Server - A Solid Foundation for SOA**

Find an Oracle Partner near you:

beloit Solutions Group

Geographic Coverage:
•AR •AZ •CO •ID •KS
•MO •MT •ND •NE
•NM •SD •WY

About Beloit Solutions Group

Beloit Solutions Group is a client-focused provider of information technology based business solutions that focus on increasing productivity and reducing costs. We believe in evolving, not revolutionizing our client systems, structure, and assets to maximize return on investment. Beloit focuses on IT Strategy and Planning, Business Process Management, Service Oriented Architecture, Application Performance Management. Portals/Corporate Communication and Identity/Access Management. By staying current on the latest innovations to solve tough business problems and using a flexible approach to our methodology, we are able to design and implement successful strategies for our clients.
www.beloitsg.com

idhasoft Vision Packaged. Delivered

Geographic Coverage:
•AL •CA •CT •FL
•GA •MA •ME •MS
•NC •NH •NV •NY
•OR •RI •SC •TN
•UT •VT•WA
•Canada

About Idhasoft

Idhasoft is a software products and IT services provider serving enterprise customers. We offer solutions to industry segments in the United States, Europe and Asia. We provide services to industry segments including Banking and Financial Services, Healthcare, Insurance, Manufacturing, Public Sector, Retail and Telecommunications. With a reputation for unparalleled service, Idhasoft offers a comprehensive range of technology solutions. From software application development, strategic IT consulting and recruitment process outsourcing to enterprise-wide implementation of third-party vendors, Idhasoft is second to none as the preeminent, complete IT partner. With expertise in robust technologies, such as Oracle and SAP, Idhasoft excels at customer satisfaction, always completing projects ahead of schedule and under budget.
www.idhasoft.com

Geographic Coverage:
•IA •IL •IN •KY
•MI •MN •OH •WI

About PSC Group, LLC

PSC Group, LLC is an information-technology and professional services consulting firm specializing in providing business process solutions to clients that view IT as a strategic resource. We partner with IBM, Microsoft, Oracle, and other leading providers (including Open Source), to deliver Agile PBM, back-end application and ERP integration, middleware, portal, and collaboration-based technologies and architectural services for the Services, Insurance, Manufacturing, Distribution, and Financial Services sectors.
www.psclistens.com

Geographic Coverage:
•LA •OK •TX

About TSA

TSA was founded as Technical & Scientific Application, Inc. in 1986 to serve the Oil & Gas and engineering community as an exclusive supplier and a repair facility for Hewlett-Packard equipment. As a trusted HP Oracle - Solution Elite Partner, TSA is a full service business partner, providing the complete product offering of HP equipment and business solutions. Our goal at TSA is to leverage our 23 years experience and our strong partnership with Hewlett-Packard and Oracle to develop long-term relationships with our clients by delivering innovative and reliable IT solutions.
http://www.tsa.com

Geographic Coverage:
•DE •MD •NJ •PA
•VA •WV

About Turnberry Solutions

Turnberry Solutions is an IT Professional Services firm with a proven track record of delivering large-scale Portal, BPM and Software Integration projects. Over the past decade, we have assembled a team of distinguished engineers and project managers who provide services in the following areas:

- Business and IT Strategy
- Requirements Analysis and Project Management
- Software Architecture, Design and Development
- Quality Assurance and Operational Support

http://www.turnberrysolutions.com