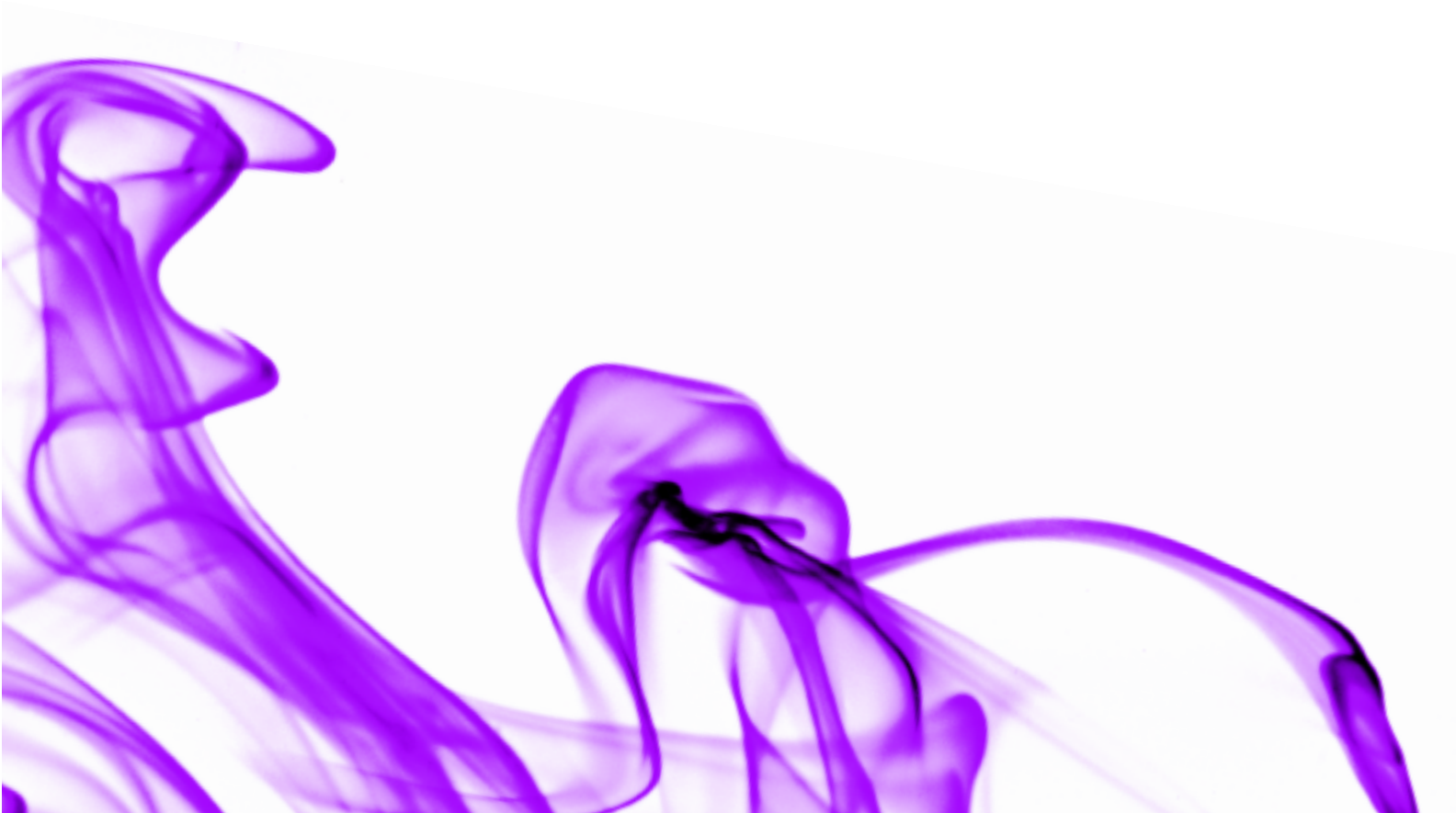
BEA White Paper

BEA AquaLogic™ Product Family

A suite of Service Infrastructure products for successfully deploying Service-Oriented Architecture in heterogeneous environments.



Copyright

June, 2005

CWP0949E0605-1A

Table of Contents

Overview

Enterprises are looking to find new and cost effective means to leverage existing investments in IT infrastructure as well as incorporate new capabilities to improve business productivity. As a means to improve the integration of applications hosted both internal and external to the enterprise, enterprises are now turning to Service-Oriented Architectures[1]. Service-Oriented Architecture is an IT strategy that organizes the discrete functions contained in enterprise applications into interoperable, standards-based services that can be combined and reused to more quickly adapt to meet dynamic business needs.

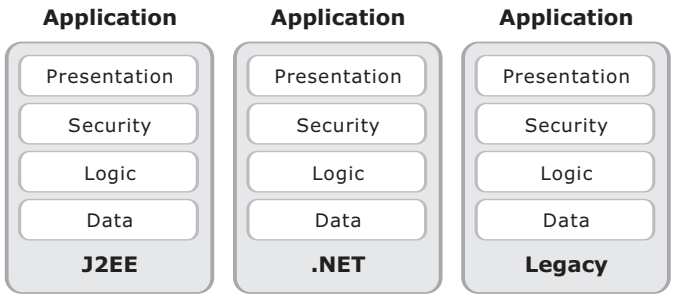
In this paper, we describe the major aspects associated with the introduction of a Service-Oriented Architecture and the impact that it can have on an enterprise's information architecture. We outline the concept of exposing data sources as services and discuss the critical integration aspects that need to be addressed including data access, data transformation, and integration into an over arching enterprise security scheme. The paper suggests alternatives, utilizing a Service-Oriented Architecture approach, to promote flexible, extensible, and evolvable information architectures.

Introduction

Business remains a very liquid environment where reacting to changes in competition, market dynamics, and regulatory mandates are critical to being successful. Information has become a strategic asset of a business, since without timely information it is difficult for a business to function effectively. Even government has realized the critical nature of information in order to effectively execute a mission whether military, humanitarian, or intelligence. With such demand for information, an enterprise's IT group is being forced to find new and cost effective means to obtain the information. Unfortunately, the budget of these groups has not been increased over the last few years leaving them with the daunting task of having to leverage existing IT assets to create the new applications required to satisfy the increasing thirst for information.

Adding to the complexity of quenching this information thirst is the fact that the necessary information is not all stored in a single data store, such as a database, but is instead stored in individual silos from which each application must drink. Each silo has its own characteristics. Some of the silos are packaged applications such as Customer Relationship Management (CRM) or Supply Chain Management systems; some are files in a file system containing text, images, and video; others are in some form of storage such as LDAP directories or databases. But each of these silos provides its own storage mechanism, its own storage format, and its own storage access mechanism.

In fact, it is seldom that a single vendor is used for each of the silo implementations, thus contributing to the situation where information remains locked up inside the silo. As a result, each application must first understand where a particular piece of the information is located; second, understand how to access those pieces of information; third, understand how the information is related; and finally, transform the various pieces of information into a unified view of something as basic as a Customer or Order.



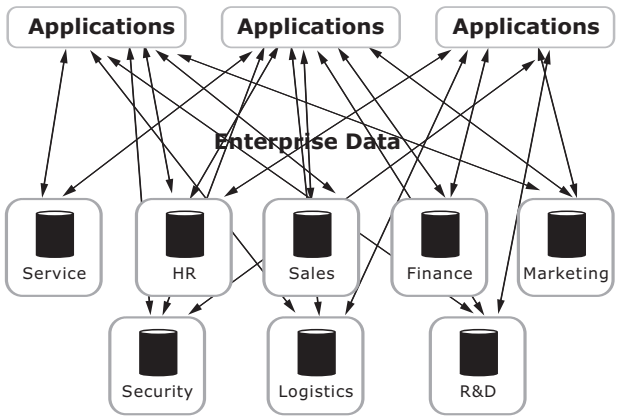
In the typical application-centric enterprise, information is locked up inside application and data silos making it difficult to access.

In an attempt to bring some order to the sea of information chaos, many enterprises are turning to a Service-Oriented Architecture approach to integration. Using this architectural approach, the capabilities of applications and the access mechanisms to information sources are exposed as services, hiding the underlying implementation from the application developer. Often the information itself is exposed as XML documents which provide structure to the often unstructured world of files contained in a file system. In addition, this architectural approach also allows the incorporation of services offered by external service providers to be included as part of applications. As a result, enterprises are able to unlock key information in a cost-effective manner while also providing them with the flexibility required to evolve as business demands change.

Data Sources as Services

In most enterprises there are a vast number of different information sources, including databases, text files, images, video, directories, and many more. Unfortunately, the number of formats in which data is stored is as varied as the types of information stores themselves. In order for an application to deal with the variety of information sources as well as their associated means of exposing information, an application developer is forced to code specifically to each source.

As each information source evolves and new information sources become available, the application must be modified to adapt specifically to the format and exposure mechanism of the information source. Although SQL provided a more general purpose mechanism through which to obtain data stored in databases, its adoption as the means to obtain information from other information sources never came about. Enter the concept of exposing information sources as services.



Integrating to each data source is time-consuming, costly, and inflexible.

With the introduction of Service-Oriented Architecture, enterprises are now creating service-based information sources. This concept focuses on hiding the implementation and access mechanism details of a given information source behind a service-based interface and away from application developers. While a Service-Oriented Architecture approach could be done with any number of technologies, the most prevalent form is the use of XML and Web Services. Using this approach, it is now possible to hide whether the information source requires a SQL query or some other mechanism to access the information locked within. Application developers no longer have to focus on learning multiple access schemes, but rather are focused on making service requests and handling service responses.

The use of technology such as WS-Security allows the application developer to be abstracted from the actual details of authenticating to the information source and dealing with connection management. But there is still a catch to all this abstraction: the vast majority of today's information sources are not service-enabled. As a result, enterprises are now faced with the task of service enablement. The two most common approaches used for service enablement are the creation of service proxies and service adapters.

Service Proxies

A service proxy is the most common form of service enablement due in part to the lack of appropriate integration points in most information sources. The concept of a service proxy is to create an external wrapper which is installed in front of an information source, such as a database, and act as a gateway or "proxy" through which access to the information source is performed. A service proxy for accessing files in a file system could be created to access a file anywhere in a file system or be restricted to only certain areas of the file system.

Service proxies are used to hide the details of what a requesting entity or "consumer" is required to do in order to establish a session with the information source, transform the service request into the appropriate set of interactions in order to retrieve or update the information service, and then format the information to be returned to the consumer. In more advanced situations, the service proxy may hide the additional steps required to determine the actual results to be returned. An example of such a requirement would be the retrieval of additional meta-data, such as data sensitivity labels that may be stored in yet another information store, that need to be retrieved and evaluated before particular portions of the result set are to be returned.

Service Adapters

Service adapters are an embedded or integrated form of a service proxy. The core difference between these two approaches is that a service adapter is integrated into the request/response processing logic of an information source, typically application packages or an application server on which the application is hosted, rather than externally to the information source.

One advantage of a service adapter is that they typically do not require the use of an additional connection between the service enablement piece and the information source itself. Unlike the service proxy model where there is typically a network connection between the consumer and the proxy, as well as a connection between the proxy and the information source, the service adapter model typically only requires support for the connection between the consumer and the adapter. This typically creates a more secure environment since communication with the information source is directly mediated by the adapter which is inter-positioned within the request processing logic of the information source.

Additional Responsibilities

But service proxies and adapters must also take on other responsibilities beyond data retrieval. Some of these responsibilities are driven by the actual requirements of the information source itself, while others are driven by external forces.

Identity and Authentication

The compliance requirements of government regulations such as the Health Insurance Portability and Accountability Act (HIPAA) and Sarbanes-Oxley can complicate the reporting process. These regulations typically require that organizations track and audit the identity of information requesters as well as the information they receive.

Many widely used information sources, including databases, directory servers, and packaged applications, use a connection-based conversation model similar to that found in client-server computing. In this model, the identity of the requester is regarded to be the same identity used to make the connection, for both authentication and audit reporting purposes. Distinguishing between the two identities would require that every time an individual identity makes a request, the service enabler must create a separate connection to the information source. Because these connections consume resources and impact performance, many applications just utilize a connection with a single identity for grouping information requests. As a result, the security mechanisms provided by the information source cannot be correctly utilized nor can the regulatory requirements be met.

This practice also places the burden of obtaining the appropriate security credentials for the requesting identity either on the requester or on the service enabler itself. While standards such as WS-Security allow for obtaining security credentials that represent the identity of the requester, the mechanism used to authenticate the requester to an application is often not the same mechanism required by the information sources used by that application.

To address these issues, a service enabler needs access to a credential exchange service. This uses the requester identity to obtain the appropriate set of credentials for authenticating with a given information source.

One commercial offering to provide a credential exchange services is BEA's AquaLogic Enterprise Security product. Provided as a service-based security infrastructure, it contains replaceable implementations, thus allowing alternative implementations including the proposed WS-Trust specification[2].

Utilizing a credential exchange service to obtain the appropriate credentials, a service enabler can support a per-identity connection model, a per-request propagation model supported by a few information sources today. It can also support identity-based connection pools, without ever exposing the actual mechanism to the requester of the service.

Services Are Not Enough

Although exposing information sources as services provides a number of benefits, it does not address a number of issues such as unified views and data transformations. Utilizing a typical web based application, let's examine some of the other aspects that need to be addressed in order for an information architecture to be flexible, extensible, and easy to evolve over time.

In a growing trend, portal applications are being used as the presentation service layer of web based applications. In this role, the portal application acts as a natural data aggregator by pulling information from a number of back end information sources and assembling generated pages. As such, developers of portal applications have been required to understand each of the information sources available, what information pieces are stored in each source, the relationships between those pieces of information, and the necessary transformations required to create a single virtual view of some business aspect such as a Customer or Order[3]. This practice provides justification for the use of building data sources as services, as mentioned above.

The result of this approach is that the knowledge and process of building the unified information views is contained in the portal application and cannot be easily shared. Furthermore, as the enterprise evolves due to a changing business climate, adaptation to this evolution results in coding changes to the portal application that must be repeated potentially in multiple places.

Unified Information Views

Exposing information sources as services, by itself, does little to address the issue of unified views of information especially when the information is contained in different information sources. Nor does it directly help with the issue of updates to the information which makes up the unified view. The part which is missing from the pure service-based approach is the logic which understands which information sources contain the information necessary to create a unified view of an entity and the relationships between the various pieces of information contained in each information source that are used to compose the unified view.

Armed with this information, an enterprise could task a developer to piece together an information service to create such a unified view. This could easily be a daunting task since the developer would have to understand each of the information sources. Complicating this issue is the requirement to perform updates to the information contained in a unified view. Typically, a developer would be faced with the issue of keeping track of where information came from and then handling the problem of knowing which piece of the information had been changed and the corresponding information sources that would need to be updated. Furthermore, the developer would also have to deal with the issue that can occur when one or more of the data stores can not be updated.

BEA's AquaLogic Data Services Platform is an example of a product that targets the issue of creating a unified information view and addressing the issue of providing the updates to the appropriate information sources when changes are made. It provides a service-based view to application developers without the need to develop the actual code to assemble the unified view. Instead, these types of products utilize graphical modeling tools which can discover and display schema contained in structured storage and allow unified views to be graphically constructed and then code generated. Furthermore, the generated "code" can be in the form of a declarative query language, such as XQuery.

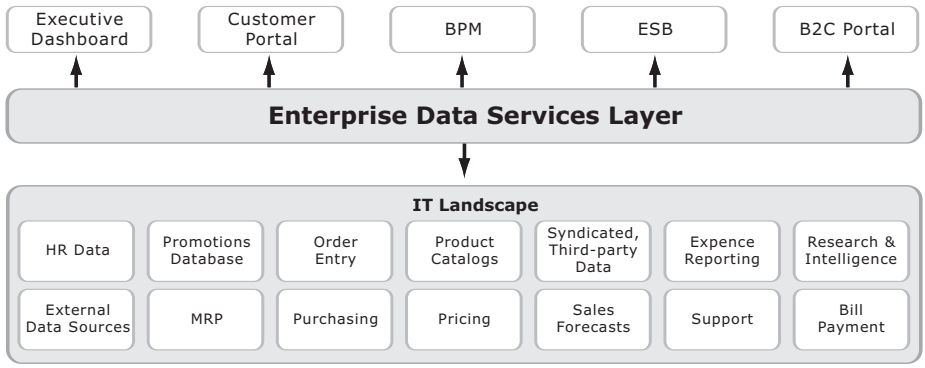
This enables a variety of optimization techniques to be brought to bear when processing service requests, particularly if—as allowed by Data Services Platform—the services are defined in a layered manner. The Data Services Platform also includes features which handle the issue of utilizing the appropriate authentication mechanism and connection management of each information source so that the identity of the requesting entity is available to the information source.

Data Transformations

Although exposing an information source as a service helps to ease application construction, this alone does not address the issue of transforming information from one format to another. Often information about a business entity, such as a Customer, Order, or Employee, will have elements that are referred to by different identifiers in each information source.

It is not necessarily practical or possible to normalize the information models of all components of an application, especially when some application components are services provided by an external partner. As a result, data transformation is required even when information sources are presented as services.

One approach to this problem is enabling each information service to normalize the information into a canonical format prior to providing it. However, most enterprises are unable to define a single data dictionary or to force their partners to utilize an enterprise's definitions.



A Data Services Platform empowers IT to quickly meet the needs of changing business goals while delivering faster time-to-value.

With this in mind, another approach is to construct infrastructure services which provide data transformation capabilities. While this approach does allow for the separation and reuse of data transformation services, it can result in the development of hard coded process flows that may also need to be changed as an enterprise's requirements evolve. These hard coded data transformation flows are a reasonable approach when the service is directly responsible for the exposure of the information source as a service, but should be avoided in other situations.

Process Integration and Services

Services that expose a unified view of information are only part of the solution. How these services are used in the creation of a business solution requires that these services be combined with other services to fulfill a predefined business process. The business process, which describes steps required to implement the process, can now utilize the information exposed as services in performing a business function. This provides a layer of abstraction that shields the consumer, who is requesting the process to be performed, from the actual steps and services used to fulfill the request. As a result, the implementation of the business process can be changed without affecting any consumers of that process.

Providing this type of business process workflow has traditionally been accomplished by describing the actual steps of the process in source code, resulting in a "hardwired" approach. As a result, each time the business process changed, there was a need for source code to be created or modified to implement the new process. The need to "hardwire" the workflow of a business process into source code has been eliminated with the introduction of standards such as the Business Process Execution Language for Web Services (BPEL)[4]. This standard provides an XML syntax for describing the steps of a business process that is made up of services and the exceptions that will need to be handled. Such an approach allows a business process to be changed without requiring source code changes, resulting in a faster time-to-value for an enterprise.

Security

Information is only as trustworthy as its pedigree and integrity. Although many information source vendors focus on secure communications, authentication, and authorization, there still remain other issues that must be addressed in the current business climate. As mentioned earlier, new regulatory statutes, privacy laws, and other business or mission requirements are now a critical component of virtually all enterprises. Compliance is no longer optional, but now contains stiff financial and personal penalties to the executives responsible for these enterprises.

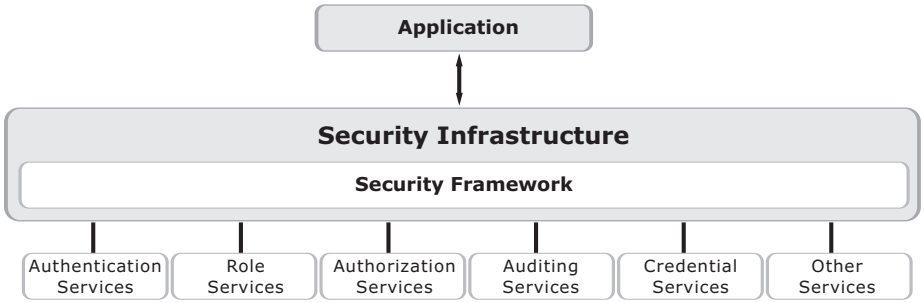
Enterprises now consume information in many different formats and from many different sources. As a result, it is difficult to store all forms of information in a single form of information repository. This is the situation that many enterprises find themselves in today: many different security solutions, none of which are integrated, and none of which can be centrally administered.

Integration with Enterprise Security

One problem that has plagued enterprises is the requirement that each application component requires its own security administration. This is a direct result of each piece of the enterprise at one point in its life, thinking it was the center of the universe. If one focuses just on information sources, one sees that each source has its own authentication mechanism, each its own authorization mechanism, and each its own auditing mechanism. As a result, significant operational cost is spent in administrating each of the individual pieces.

A number of the security vendors have each attempted to address this problem through the introduction of centralized policy servers. In addition, a number of information storage vendors have attempted to use the security in their storage system as the authoritative source for policy decisions. Unfortunately, these approaches have not managed to scale well, especially when used in a highly distributed environment such as a Service-Oriented Architecture. To illustrate this point, a recent pilot of the United States Department of Defense's Network Centric Enterprise Services (NCES), showed that the network latency alone associated with making remote authorization decisions resulted in 70% performance degradation for each Web service business request[5]. Although additional bandwidth would address some of the performance issues, it would not address all of the network latency issues, or the potential for the security services to be a bottleneck or single point of failure.

In keeping with the security best practice of protecting a resource, such as information, as close to the resource as possible, enterprises are faced with how to unify the protection without suffering the performance and scaling issues. BEA's AquaLogic Enterprise Security offering provides a set of security services which allows enforcement to be performed as close to the resources as feasible. Like any other service, the implementation details of each security service are abstracted through a service interface. The security infrastructure provided through this approach can be utilized in both the service proxy and service adapter approaches, as well as within applications themselves.



A services-based approach enables your applications to leverage a common application security infrastructure across the enterprise—to reduce costs, strengthen security, and accelerate time-to-value.

To efficiently administer a distributed environment requires a centralized administration approach. This approach must also be built on services so that different implementations can be integrated allowing existing investments to be incorporated. Emerging standards around provisioning and policy language, such as Service Provisioning Markup Language (SPML)[6] and the XML Access Control Markup Language (XACML)[7], respectively, are the building blocks on which many of the security silos found in today's information sources can be integrated.

Putting it All Together

With a set of services and data transformations created, along with a representation of how the services will be used to satisfy a business process, and a strategy for enforcing security, its time to consider how a consumer (a client) or service acting as a consumer will physically interact with another service, acting as producers, in order achieve the business goals. It's typically at this juncture that one realizes that not all services are implemented as Web services using SOAP over HTTP and a synchronous invocation style of programming. In fact, some of the services are long running and thus utilize an asynchronous invocation style and may only be available over transports such as FTP, SMTP, and even message queues. Faced with this situation, service consumers must make a critical choice; one that could result in tight coupling with the service implementation even though it was designed as a service. Service consumers must choose to either code to the invocation style, transport, and security or look for another alternative.

The Service Neighborhood

One approach is to utilize an intermediary, such as a service broker, to create a mediation layer between the service consumer and the service producer. Through the use of a service broker, the service consumer can become loosely coupled to any service producer. This is because a virtual service endpoint is created for each of the services attached to it. Each virtual service endpoint can abstract the differences in invocation style and message transports from both parties. Finally, the service broker can also provide the execution of the business process flow or pipeline required to satisfy business requirements. Through this, it is possible to create a loosely coupled environment where service consumers, data transformations, business process workflow, and service producers can operate independent of one another.

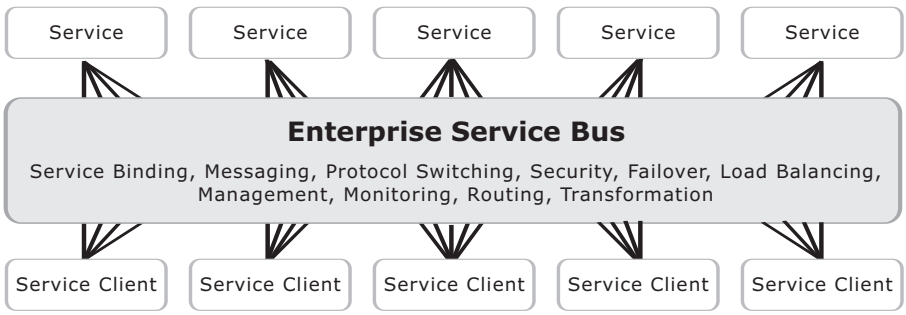
This kind of topology is ideal for the creation of an initial neighborhood of service consumers and service producers. The broker acts like the roads throughout a neighborhood allowing a service request to find its way from a service consumer to a service producer. Service brokers are typically appropriate for use where a smaller population of service consumers and producers are found. Their routing, pipeline, and data transformation capabilities are scoped to those services that are directly brokered. As the population of service consumers and producers begins to grow and expand to greater scopes, its time to look to an Enterprise Service Bus.

Enterprise Service Bus: The Service Interstate Highway

As services proliferate throughout an enterprise, the number of point-to-point connections utilized will quickly become unmanageable as composite applications are developed and deployed. Additionally, business process definitions will become embedded in application logic making it difficult to adapt quickly to change in the business environment. Couple this with the fact that service consumers are forced to address the incompatibilities introduced by services utilizing different transports, portals, and interaction styles, and one has a formula for failure. What is needed is a means to abstract these items away from the developer.

An Enterprise Service Bus or "ESB" is a critical component that simplifies the integration and flexible reuse of business components using a service-oriented architecture. An ESB provides the enterprise orchestration layer for the distributed processing of service requests. An ESB provides an intermediation approach which abstracts the interaction details between a service consumer and one or more service producers. An ESB can provide the flexibility to transparently interconnect services allowing even richer applications to be composed.

Through the use of an ESB, such as BEA's AquaLogic Service Bus, enterprises can now quickly assemble applications from any of the services which are "plugged" anywhere in the bus to create new composite applications. One of the key values of introducing an ESB is that it removes the need to hard code a particular processing flow of services that spans more than a single bus segment, by allowing the processing flow to be described and then dynamically assembled using a configuration-based (as opposed to programming-based) paradigm. Through the orchestration of services, it is now possible to dynamically change how the IT infrastructure utilizes information sources, performs the necessary data transformations, and incorporates capabilities provided by and to external partners to allow an enterprise to more efficiently execute its goals.



An Enterprise Service Bus (ESB) delivers the service integration foundation for IT agility and cost-effective alignment with business needs.

BEA's AquaLogic Service Bus integrates each of the service neighborhoods, into an overall multi-segment bus of services. This bus of services extends the ability to allow data transformations and other services to be "plugged" into the bus of services and then reused by any number of different services and application components throughout any segment of the bus. In addition, Service Bus extends the message-level security across segment boundaries to allow identity and the integrity of service messages to be maintained throughout each of the segments of the bus.

The result of utilizing BEA's AquaLogic Service Bus is that service consumers can remain unaware of the location of the service producers as the services in the enterprise continue to proliferate. In addition, the mechanisms used for the delivery of service requests are enhanced to take into consideration the best route to a service producer across the various segments. This is critical since there may now be different versions or instances of a given service producer available in the ESB.

Although exposing information sources as services, providing the appropriate data transformations, utilizing business process workflow definitions, and connecting services together to create composite applications may seem sufficient, its actually lacking a number of critical items that are necessary to allow the solution to grow to an enterprise level that can be efficiently managed to create a greater time-to-value. An ESB assists in addressing some of these issues. However, it still does not address gaps in security and management visibility. Which can occur when service consumers and providers are not integrated with the service bus. Enter Service Infrastructure.

Service Infrastructure: The Whole Is Greater

Service Infrastructure is a unified approach to services management that combines the capabilities of an Enterprise Service Bus, business process management, and services, with proven enterprise-wide approaches to create a fabric of services for constructing and deploying new composite business applications. A Service Infrastructure extends the capabilities of an ESB to provide enhanced service virtualization, allowing multiple instances of a service producer to be considered a single "virtual" producer, even if the multiple instances are on different machines or are implemented differently. This is possible because a true Service Infrastructure abstracts away the notion of individual providers by providing a single (albeit virtual) location for message transformation and routing to happen, all in a secure manner.

A Service Infrastructure also enhances the layer of intermediation provided by the ESB to provide intelligent routing of service requests, including the ability to take security into account when making decisions on routing or aggregating data into specific views, as well as the ability to provide additional SLA status information that is used in determining the best service endpoint to which the service request should be routed. This is a concept that goes well beyond what most Enterprise Service Bus products deliver today. As a result, many of them miss one crucial fact about building out a Service-Oriented Enterprise. As services begin to accrete in an enterprise, there is an increasing need to have a management framework that provides visibility into a number of key indicators in order to ensure that service-level agreements can be maintained and the lifecycle of services can be managed. Visibility into the performance of a service is not only critical for assessing service-level agreement, but is also a critical set of criteria that can be used to more effectively route messages to service instances. Along with routing information, a Service Infrastructure must consider the security implications of delivery to an inappropriate service end point.

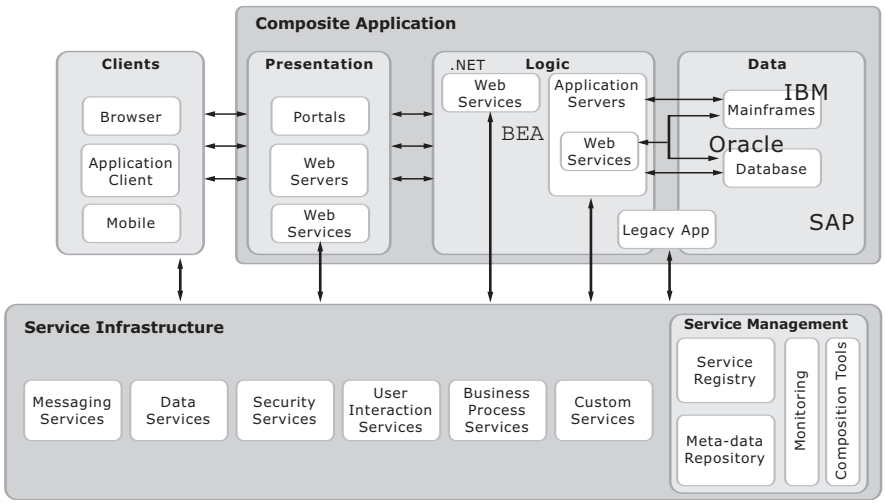
In order to accomplish this, a Service Infrastructure needs management points that are as near to the service, if not embedded, as possible. Using the same reasoning as placing security as close to the resource to be protected as possible, the ability to observe all of the service requests being handled by a service virtually eliminates the possibility of there being unseen traffic handled by the service. As a result, better decisions concerning the delivery to a specific service can be made. This is analogous to driving down a major interstate highway in a large crowded city at rush hour. Motorists typically tune in to a traffic radio station to not only find out the conditions on the interstate highway but also at the off ramps and city streets so that corrections can be made early enough to ensure an on-time arrival.

In order to understand the value of a Service Infrastructure, consider the following example. When someone has a minor accident, like breaking a leg, they go to the hospital. The hospital may need to access multiple data stores for information on that person. The hospital will have billing information, prescription information, and patient history records. This data will all need to be aggregated into a "single view" of the patient. The emergency room physicians will be able to call up the patient's record, but due to privacy rules, may not be able to see all of the patient information. The data will be filtered and presented according to appropriate security policy.

Now the hospital must get paid. In order to do this, the hospital's notion of "patient" must be transformed into something the insurer can understand. This information must be transformed into a "policyholder" and then routed to the insurance company.

The example illustrates a use-case which shows how each of the three key parts of the Service Infrastructure comes into play. First, Data Services are used to provide a single view of an enterprise's data. Second, Enterprise Security is used to make sure the appropriate security and privacy rules are followed and tracked. Finally, the Service Bus transforms the data and routes it appropriately.

When all of these components are brought together in one product family, a significant evolution in the creation of enterprise-wide applications can be achieved. With these technologies in place, one can now turn their attention to the actual creation of applications composed of services as well as their management.



Service Infrastructure: The whole is greater than the sum of the parts.

Composition of Applications

One of the main value propositions for using a service-oriented approach to application development is the increase in component reuse and the reduction in development times. Service Infrastructure replaces the programming-driven model with a metadata-based application framework that significantly reduces development cycles. The framework enables services to be assembled according to business process models so that composite applications can be swiftly created, not only by developers, but also by application specialists or business analysts able to translate business process, data, security and user interaction requirements directly into an application definition.

In the future, BEA plans to deliver a unified composition environment that will provide the basis for composite application assembly from services that are deployed throughout the Service Infrastructure. This unified composition environment will provide a business-centric presentation that allows users to view available services, construct composite applications utilizing business process model definitions in context with services, as well as manage and monitor deployed services. In addition, it will be extensible allowing users to add tool extensions to support the seamless use of third-party applications and services in tandem with the BEA AquaLogic product family.

When BEA introduces new products to the AquaLogic product family or when an enterprise chooses to add new products to its Service Infrastructure, BEA's integrated composition environment will incorporate and expose these new capabilities within the appropriate business context. As a result, users will be provided with a role-based, business-centric view of the enterprise and will not be required to utilize separate administrative or application consoles for each AquaLogic product or third-party application added to the Service Infrastructure environment.

Management of Services

Management is another critical component of a successful Service Infrastructure. Because of the distributed and heterogeneous nature of a Service Infrastructure, it is not cost effective to attempt to manage each service individually. And as an enterprise expands its use of services, each of the individually managed ESB segments needs to be incorporated into a unified management infrastructure in order to reduce the cost of operation.

The AquaLogic product family accomplishes this through its management framework that supports the management of a distributed environment from a central point of administration. Utilizing a web based graphical interface, management of particular ESB segments and service endpoints can be partitioned so that administration can be delegated to different individuals. The AquaLogic management framework provides programmatic support by exposing its management capabilities as a set of web services. This provides the ability to integrate the management of AquaLogic products into existing enterprise management consoles as well as scripting with any web service enabled scripting language.

Service deployment and un-deployment, service and infrastructure configuration, security policy, business process definitions, as well as other information is centrally administered and distributed to the appropriate locations in the Service Infrastructure. Distribution of such information needs to be performed in a transactional manner to ensure that the integrity of the Service Infrastructure isn't compromised due to interruptions of service at any point in time. As an illustration, consider the following scenario. An enterprise has deployed a set

of composite applications which include a set of services and a business process. Due to changes in the business, it is now necessary for the business process to change and requires the use of an additional service. Clearly, the new service must be deployed before the updated business process can be utilized. But doing so can be a labor intensive task, especially in a highly distributed enterprise. As a result, the deployment of the updated business process must be done in a sequence to ensure that the integrity of the existing system is maintained. Should a failure in the deployment of the new service not be noticed, it would be possible for new business process deployment to result in failure of all business transactions.

Conclusion

The concept of exposing information sources and applications as services provides a number of benefits beyond the standard approach to information retrieval, application integration and user interaction. The abstraction of retrieval mechanisms, connection management, and security, along with the presentation of information in XML provides application and service developers a consistent approach regardless of how the underlying information source is implemented. Additional strides still need to occur in order for information sources and applications to more naturally and easily integrate into a service-based enterprise. These include the ability to expose information access mechanisms as services native to the information source, the ability to propagate identity and associate it with a request without significant resource consumption, and the ability for the security enforcement provided by information sources to be incorporated into an enterprise security and management infrastructure are still obstacles to be overcome.

The basic concepts of building applications and exposing them as services have been widely used as the initial step in implementing a service-oriented architecture. Service Infrastructure extends these initial concepts by adding dynamic data transformation and message services that can be applied to composite applications and services within context of consistent enforcement of enterprise security policies, data access privileges, business processes, and user interaction. All of these services will be brought together within a unified composition and management environment that will allow application specialists and business analysts to compose and manage business-centric applications to support the goals of the enterprise.

Notes:

1 A version of this paper previously appeared in the proceedings of SIGMOD 2005, June 14–16, 2005, Baltimore, Maryland, USA

2 Web Services Trust Language (WS-Trust), Feb 2005, OASIS

3 Chappell, D., Enterprise Service Bus, O'Reilly Media, Sebastopol, CA, 2004

4 Business Process Execution Language for Web Services, Version 1.1, May 5, 2003, OASIS

5 Fusion FY2004 After Action Report, Nov 05, 2004, Department of Defense, Assistant Secretary of Defense for Network and Information Integration/DoD CIO, pp 53-55

6 Service Provisioning Markup Language, Version 1.0, June 3, 2003, OASIS

7 eXtensible Access Control Markup Language (XACML), Dec 6, 2004, OASIS

About BEA

BEA Systems, Inc. (Nasdaq: BEAS) is a world leader in enterprise infrastructure software, providing standards-based platforms to free the flow of information, services, and business processes. BEA product lines—WebLogic®, Tuxedo®, and the new AquaLogic™ family of Service Infrastructure products—help customers reduce IT complexity and successfully deploy Service-Oriented Architectures to improve business agility and efficiency. Headquartered in Silicon Valley, BEA is a billion-dollar company with 15,000 customers worldwide served by 76 offices in 36 countries. More information at *bea.com*.

BEA Systems, Inc.

2315 North First Street
San Jose, CA 95131
+1.800.817.4232
+1.408.570.8000

bea.com

CWP0949E0605-1A