# SOA BLUEPRINT – REFERENCE ARCHITECTURE
## VERSION 1.1

**SOA Alliance**
Group of SOA Practitioners

**Abstract**
*Service Oriented Architecture (SOA) is the business operations strategy for leveraging information to meet organizational objectives such as growing revenue, increasing customer satisfaction and improving product quality.*

*SOA is not a well traveled road and lacks many of the shared experiences, assets and patterns required for widespread and reliable adoption. Moreover, without a common language and industry blueprints, SOA may fail to deliver the promised benefits of intra and inter-enterprise services reuse and process interoperability – instead adding more custom logic and increased complexity to IT infrastructure..*

*A group of SOA Practitioners have agreed to come together under the SOA Alliance to provide leadership in the industry to address the challenges. The SOA Blueprint is envisioned as a multi-volume collection of publications that can act as a standard reference encyclopedia for all SOA stakeholders. This document, the SOA Reference Architecture, is an early asset created as part of the broader SOA Blueprint initiative. It is intended to provide the end user / consumer perspective which hopefully will influence both the vendor community and the standards organizations.*

**Keywords**
*Service Oriented Architecture, SOA, SOA Alliance*

## Introduction

Today, there is a lot of hype around SOA and it is expected to continue until the industry matures. SOA is not a well traveled road and has the potential for failing to deliver the promised benefits of intra and inter-enterprise services reuse and process interoperability. Instead, it may add more custom logic and propagate the IT legacy. Following are some of the reasons for concern that SOA may not deliver on its promises.

- Every major vendor claims to have adopted SOA and have published their own view and reference architecture around SOA.
- Every major standards body has multiple working or expert groups attempting to define the SOA Blueprint or SOA Reference Architecture from their point of view.
- Even though SOA is in the initial phases, there are not sufficient development and management tools.
- Enterprises are attempting to solve similar problems but without a forum for sharing best practices across the industry. Various product vendors, system integrators, analysts have all attempted to share these practices – however, the information may have been lost in translation because of lack of common vocabulary across the industry.

These conditions add to the confusion resulting in delayed adoption of SOA.

### SOA Reference Architecture – Definition

The SOA Reference Architecture is an ideal "Target State" architecture for an Enterprise or Line-Of-Business (LOB). Some also refer to this as the "Future State" or "Future Vision" of the Enterprise. The objective of the SOA Blueprint is to provide enterprises the ability to build a Roadmap to start the journey to the Target State from their Current State.

### SOA Reference Architecture Approach

One needs to understand two aspects of SOA to be able to develop the reference architecture:

- The three SOA Foundation components that drive Service Oriented Architecture
- Enterprise SOA Maturity Model

### SOA Foundation

The SOA Foundation components are illustrated in the figure below.

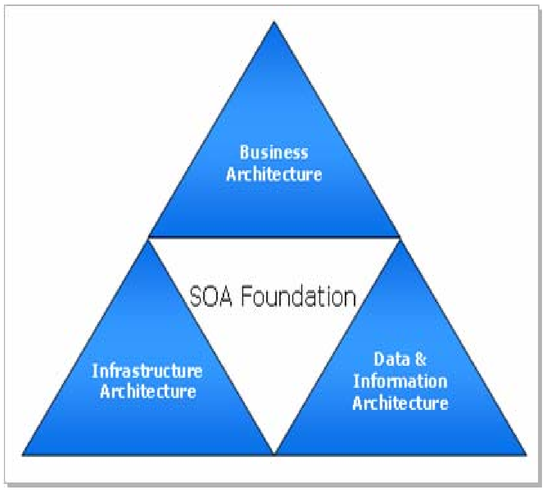


Figure 1 -- SOA Foundation

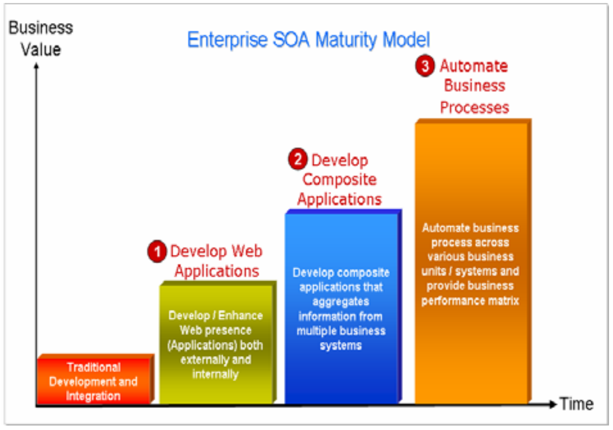
The three foundation components are:

- *Business Architecture:* Based on the business strategy, objectives, priorities and processes. Getting this right is essential for the successful implementation of SOA. One of the major benefits of SOA is reuse of business processes which provides higher ROI than the potential reuse of Infrastructure or Data components. This also includes the business processes as well as implementation of business applications.
- *Infrastructure Architecture:* This is the engine that enables SOA and should address all the aspects of the infrastructure from networks, servers, data centers, firewalls, to application infrastructure, security, monitoring, middleware, etc.
- *Information and Data Architecture:* This deals with identifying the Key Performance Indicators and the information needs that drive the enterprise. Data Architecture deals with the logical and physical modeling of the data as well as data manipulation and data quality.

The SOA Reference architecture covers each of these areas at length by providing approaches, requirements and design patterns wherever possible.

**Enterprise SOA Maturity Model**

The SOA maturity model helps enterprises develop a roadmap to achieve their Target State.



Business Value
Enterprise SOA Maturity Model
3 Automate Business Processes
2 Develop Composite Applications
1 Develop Web Applications
Traditional Development and Integration
Develop / Enhance Web presence (Applications) both externally and internally
Develop composite applications that aggregates information from multiple business systems
Automate business process across various business units / systems and provide business performance matrix
Time

Figure 2 – Enterprise SOA Maturity Model

The above diagram illustrated the enterprise SOA maturity model which can be classified into following stages.

- *Web Application Development Stage:* Provide browser based business solutions to both internal and external users. This could be in the form of rolling out web based CRM, ERP or custom applications. In addition, IT organizations would typically deploy enterprise services such as content management, search, instant messaging, discussion forums, white board, etc.
- *Develop Composite Applications:* Access and provide aggregated information from multiple sources to the users, initially internally and later externally. This generally requires focus on improving data quality.
- *Automate Business Process:* This is the stage where the applications, data and infrastructure work with the user to provide the capability that need to perform their roles effectively in the organizations. It empowers them by providing the right information at the right time. It is this stage where the enterprise matures and is enabled to achieve higher ROI by consolidating multiple business systems to a single system. This also requires business organizations to transform from their current state to the target state of end-to-end business process management, rather than point solutions.

## SOA Reference Architecture

The following diagram illustrates the SOA Reference Architecture which is categorized into three tiers – Web Application Tier, Service Tier and Application Tier.

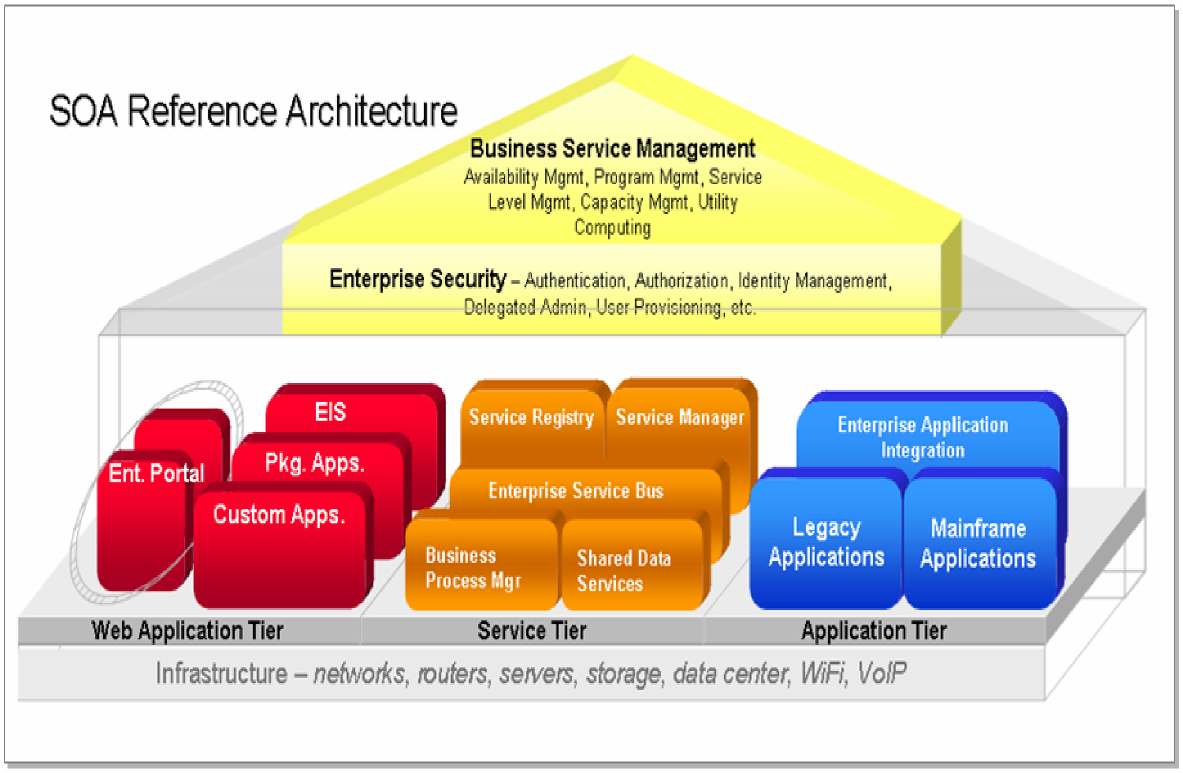

SOA Reference Architecture
Business Service Management
Availability Mgmt, Program Mgmt, Service Level Mgmt, Capacity Mgmt, Utility Computing
Enterprise Security – Authentication, Authorization, Identity Management, Delegated Admin, User Provisioning, etc.
EIS
Ent. Portal
Pkg. Apps.
Custom Apps.
Service Registry
Service Manager
Enterprise Service Bus
Business Process Mgr
Shared Data Services
Enterprise Application Integration
Legacy Applications
Mainframe Applications
Web Application Tier
Service Tier
Application Tier
Infrastructure – networks, routers, servers, storage, data center, WiFi, VoIP

Figure 3 -- SOA Reference Architecture

It is not necessary for IT organizations to deploy the entire infrastructure identified in this SOA Reference Architecture. One of the SOA Best Practices is to invest in the infrastructure only whenever it is required to provide business solutions. Following is a brief description of each of the components:

**Web Application Tier**
The primary requirement for this tier is that all the business systems / solutions should be accessible from any (supported) browser. To a large extent, this is the user interface or the presentation tier and shall contain business logic for components such as enterprise infrastructure services, applications, etc.

## Packaged Applications

Typically enterprises tend to go out in the market and license the best of the breed packaged applications that meet their businesses requirements. IT organizations, either themselves or by leveraging the System Integrators, then tailor the packaged applications to meet their needs. Examples of such packaged applications are Customer Relationship Management (Call Center, Sales Force Automation, Campaign Management, Order Management, etc.), Enterprise Resource Planning (Human Resources, Finance, Contract Management, etc.) or other industry-specific large application suites.

Most of the packaged applications are now internet protocol based which means that users can access many of its functions using any (supported) browser. Some of the latest versions of the packaged application have provided the capability to expose a limited set of functions as discrete callable services or externally controlled business processes.

Some of the best practices for leveraging packaged applications include:

- Identify and implement the best of the breed packaged applications that meet the business requirement.
- Limit the amount of custom development requirement making it easier and cheaper to maintain and upgrade.
- Attempt to achieve one standard implementation worldwide.
- Leverage the UI and the business process provided by the packaged applications, wherever possible.
- Leverage Published API's rather than directly accessing the DB.

Following are recommended approaches for taking the Packaged Application through the SOA Maturity Model:

**1. Develop Web Applications**

- Deploy the latest version of the application that is accessible by any browser; preferably a version that supports appropriate portal standards such as WSRP.
- Expose application services for consumption by Custom Applications, preferably as web services. This may require an adapter to enable access the application. Some recent versions of applications provide Integration Gateways or Web Services access directly to the application services.
- Provide seamless user experience by incorporating the enterprise look and feel (templates, skins, skeletons, CSS) as well as integrating with the enterprise Single Sign-On Solution.
- Externalize Authentication by integrating to the Enterprise Identity and Access Manger (typically LDAP).

**2. Develop Composite Applications**

- Identify business objects that could be shared across the enterprise as composite applications.
- Send event notifications (triggers) to the composite applications to initiative specific actions.
- Modify business processes and user interfaces as required to enable the composite applications.
- Expose additional business services to enable the composite applications to synchronize / update the packaged application.

**3. Automate Business Processes**

- Understand and model business processes to identify opportunities for re-engineering.
- Identify re-usable portions of business processes that can potentially be automated by a business process engine.
- Expand the number of services and business processes already exposed in the prior stage.
- Reduce / consolidate the number of applications deployed.

## Custom Applications

Organizations may prefer to create a distinct brand and unique experience for their customers and partners that is significantly different than the one offered by the off-the-shelf packaged applications. This requires providing a consistent seamless interface to the users (both internal and external). Packaged applications have the following limitations in this regard:

- Making modifications to the user navigation or user interface for some of the core transactions is not easy.
- As most of the major packaged applications are not based on open/standard technologies, their performance may not scale to the business needs.
- Proprietary development model makes it difficult to find resources or rapidly deploy new business capability.
- Integration to other technology is not straight forward resulting in point-to-point integration and possibly poor data quality.

Following are some options for developing custom applications.

1. Develop and deploy custom applications on an Application Server
2. Develop and deploy custom applications by leveraging a Portal
3. Develop a thick client either using tools based on open standards or proprietary development tools

This document shall focus on options 1 and 2. The first step for IT organizations is to determine the approach, infrastructure and tools for developing custom applications. In addition, IT organizations need to define the governance and organization model to develop the custom solution. This is not in scope for the SOA Reference Architecture document.

A short note on the thick client custom applications; these applications are typically developed using SWING, Visual Studio or similar other tools. Most of these thick clients need to interface with some external systems and the recommended approach would be to leverage open standards such as SOAP, Web Services, XMPP, WebDAV etc. instead of directly accessing any external resources such as databases, file systems or the like. This approach makes it easier for IT organizations to support and upgrade the integration.

**Custom Applications Business Requirements**

Typically most enterprises have already deployed external sites as well as multiple internal sites/applications to support the diverse needs of each of the business units. These are most probably built in silos and the first step is to standardize (unify) the look feel and the infrastructure across the enterprise which shall make it easier for a customer, partner and an employee to get the information they are seeking.

Following are the business requirements for this phase which are based on various survey feedbacks from users and discussion with various business units.

- Unify user experience on the external site, making it easy for potential users, partners, customers and analysts to find information that they are looking for.

- Standardize the look and feel across all sites (internal and external) as well as process and procedures for publishing content.
- Create one my<company name> site for all employees, contractors, partners, customers to personalize the services/content.
- Provide secure access to confidential information for all sites (internal and external).
- Provide a highly reliable, available and scalable environment.
- Facilitate branding and accessing multiple application through a common portal.
- Allow users to login once and gain access to all their services.
- Ability to personalize service based on roles and responsibility of the user.
- Reduce maintenance cost of maintaining multiple systems/applications; standardize on one platform/environment.
- Standardize on one look and feel; eliminate multiple user training requirements.
- Reduce operations and support cost to enable IT to deploy scarce resources on developing new functionality.

**Custom Applications Architecture Approach**

As Portals provide a proven set of capabilities in support of the presentation later, most IT organizations have started standardizing on a portal for developing custom applications. Following is a recommended architecture approach.

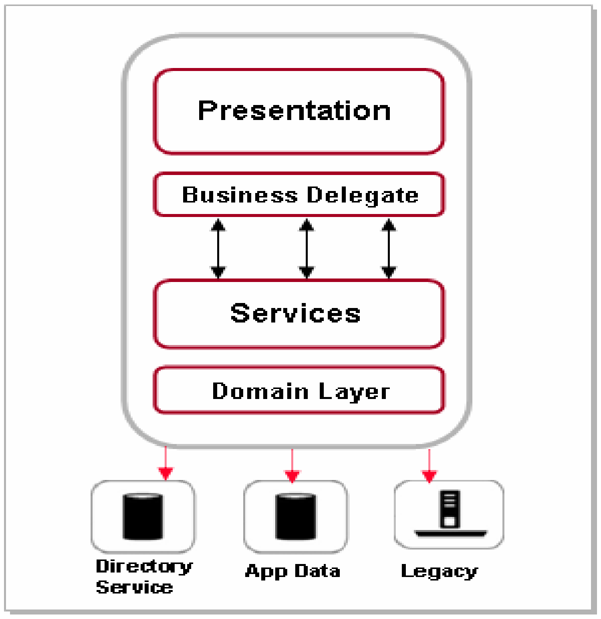


Figure 4 – Custom Application Architecture

This proposed architecture would provide the following benefits:

- Based on SOA which promotes re-use at all levels.
- Provides capabilities to deliver in weeks not months (once there is a stable framework in place).
- Leverage each product for what it is good at, example: Portal for presentation based on entitlements.
- Allow business to combine services to deliver new capabilities.
- Domain Layer abstracts the data source and the relationship, thereby minimizing the impact of changes to the source systems.
- Loosely coupling Presentation from the business logic makes it reliable and scalable.
- Consistent with SOA principles.

Following are the roles of each of the layers in the proposed architecture:

1. **Presentation Layer:** A Portal is responsible for handling all presentation services. Portlets drive the user experience where a portlet is a view on an application.

2. **Business Delegate Layer**: Components responsible for the communication between the presentation and the business layers. Business Delegates abstract the communication details and complexities involved in making a call to the business layer. It includes a Model View Controller framework that facilitates the user navigating through the web site.

3. **Services Layer:** The Services Layer utilizes the capabilities of the Application Server. It is composed of stateless functions that expose high-level business functionality. It includes a Session Façade which is the entry point to the business layer. Session Facades abstract away the details of handling fine-grained business entities from the presentation layers. Most of the business logic can be implemented directly on Session Facades or on a sub-layer commonly designated as Application Objects.

4. **Domain Layer:** The Domain Layer also utilizes the capabilities of the core Application Server. It is a collection of business entities that define persistent business concepts. Technologies that handle database storage need to be used in this layer since these components represent persistent state. Entity Beans are an example of technology

than can be used to implement some of the components of the Object Model. Alternatively Plain Old Java Objects (POJO) can be used with the help of Data Access Objects (DAO) for persistence. Entity Beans are the preferred mechanism to implement this layer but a combination of technologies may be required depending on the complexity of the Object Model.

**Custom Application Framework Components**
Custom Application Framework Components basically extend services which are inherent in the application server platform. Following are the list of Framework Components.

1. **Data Services:** This is basically the persistence layer provided for the applications. The container management is robust enough these days to leverage CMP for most of the simple transactions. It would be also be prudent to provide DAO as an option for handling any complex transactions.

2. **Logging Services:** Every enterprise should standardize the logging services used by applications and is best to leverage the features provided by JDK 1.4. Various types of message could be logged such as debug messages to trace any issues, error or fault logging for diagnostic purposes, activity logging for audit trail and usage analysis, etc. If the logging service is generic across the enterprise, it will enable the staff to more effectively determine performance or transaction bottlenecks. Logging Services involve standardizing the mechanism, communicating it to the entire development community within the enterprise, and ensuring compliance with the standard. No specific code needs to be developed for this service.

3. **Exception Handling:** This is similar to logging services in that standard application server capabilities should be leveraged. The task is to decide what mechanism to use and communicate it to the entire development community within the enterprise. No specific code needs to be developed for this service but it would be useful if examples of handling exception are also provided.

4. **Deployment/Application Configuration:** This also involves standardizing the mechanism of deploying an application in every environment, development, QA, UAT, staging and production. The document should also contain details on how to build and deploy the applications across the various environments.

5. **Monitoring:** There is a need for operations staff to monitor the platform and applications and proactively resolve issues. In addition, most external sites have an uptime requirement of over 99%. Typically all operations departments within IT would already have identified and deployed a monitoring tool. There is a need to standardize and document the development requirements to integrate with the applications or platform used by the existing monitoring tool. In some case, there would also be a need to for additional specialized monitoring tool which may need to be purchased or developed and deployed.

6. **Search Framework:** Most portal applications need to present data in a tabular format to the users. Instead of each developer attempting to resolve this problem it makes sense to develop a "search framework" which could be leveraged. The following diagram illustrates the architecture approach to the Search Framework.

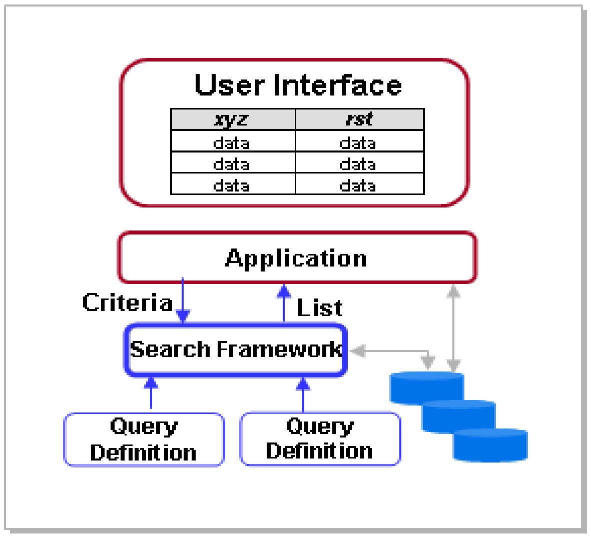


Figure 5 – Search Framework

Following are the functionality provided by the search framework:

- Dynamic query generation based on user input
  - Sort order, joins, etc.
  - Count total search results for display purposes
- Consistent mechanism for handling searches
  - Character escaping and wildcard interpretation
  - Pagination
- Abstract all database access code from application
  - Criteria used as input

    - Search results required in standards such as java.util.List
- Queries reside on external files
- Utilities to handle common UI tasks
    - Pagination
    - Criteria Persistence

7. **Notification Framework:** The objective of this component is to provide a single notification client to all applications, support Synchronous and Asynchronous interface to the Notification Engine and also provide capability to send notification through multiple channels.

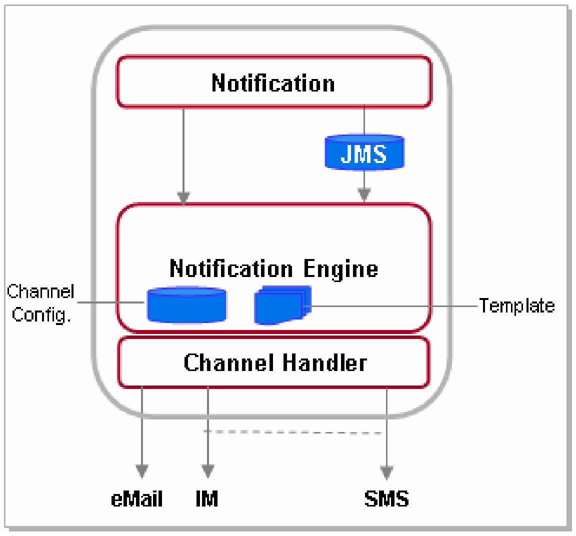


Figure 6 – Notification Framework

The interface to the various channels could be developed as required for providing the business capability.

8. **Service Proxy Framework:** Allows services to be deployed either locally or remotely without the calling application needing to know the implementation details or location of the service. The concept is very simple; the service locater determines the location of the service and calls it in the appropriate fashion. It would support multiple proxies such as EJB, Web Service and Service Bus Proxy; additional proxy types can be developed as required. This could also be leveraged by the Business Delegate to separate the presentation layer from the service layer.

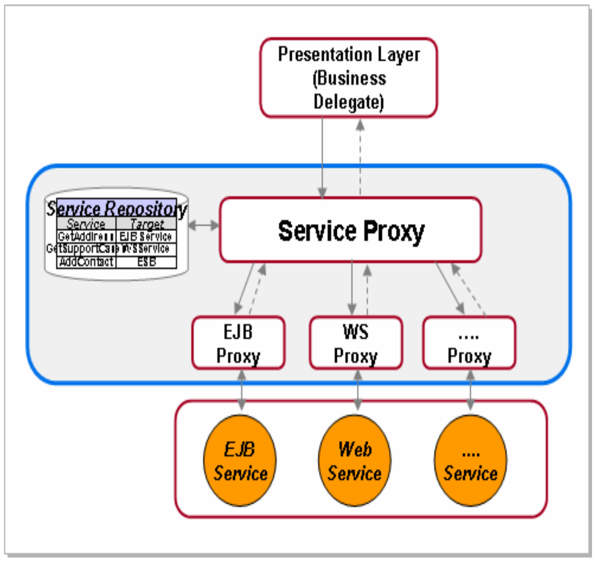


Figure 7 – Service Proxy

9. **Security Framework:** Today, most application project teams develop their own security layer, especially as the current enterprise security solutions do not meet all their business needs. There is a need to develop a security framework, that supports the client side to reduce, if not eliminate the need for developing custom security code. Following are some of those function features required to be supported by the Security Framework:

    - Single-Sign-On (SSO): capability to login once and be able to traverse from application to application without having to login again.
    - Access Control” a set of security features that addresses three main areas:
        - Authentication: determining the identity of the user interacting with the applications.
        - Authorization: determining if a user is allowed to perform a particular action.
        - Auditing: tracking the actions performed by the users.

    Several secondary services are also required such as registration, entitlement granting and entitlement querying. These features should be provided as a generic framework that can be used and reused by different applications, each with slightly different needs but all having the same basic requirements.

    - Identity Management – Typically in a large organization, there are multiple stores for managing the access control information for a set of applications / services. This may result in severe management problems. Identity Management helps by centralizing the access control management capability, as well as provisioning the users across the enterprise.

- Consolidated User Profile: This is a capability provided by the Portal to enable the application to extend the base profile. This is typically done by extracting the user profile from multiple data sources, such as the base profile and the application specific profile that is extracted from the application specific repository.
- Registration, Delegated Administration, Provisioning, Repository: These are basically security extensions built on top of Access Control to meet the application specific business needs. Alternately, these could be packaged solutions that can be easily integrated with Access Control.

Note that most of these capabilities are generally provided off-the-self by a Portal product. IT organizations will either have to develop and support this capability if they do not own a Portal for developing custom applications.

**Portal Services**

The primary function of the portal services is to manage the presentation tier of the application. As the presentation is generally based on entitlements, there is a need to support this capability.

**1.** **Presentation:** The objective is to leverage the Portal presentation capability by providing the skins/templates/skeletons/Style sheets etc. for each of the application teams. This should also include some sample applications to jumpstart the application developing, including leveraging the portal navigation capability, both for vertical navigation bars as well as horizontal tabs.

**2.** **Personalization:** Personalization services, such as portlet layout, background template selection, etc., are provided by the Portal during this phase. Additional personalization, in context to the application shall be discussed further in the profile management section of Enterprise Security.

**3.** **Authentication:** All Portal products provide this capability and the best practices are to externalize this service. Generally most, if not all, enterprises have implemented a global directory services (such as LDAP) within the enterprise. The Custom Application Framework should provide an authentication interface and externalize the service.

4. **Single Sign-On (SSO):** This enables the enterprse to provide a seamless user experience by not requiring multiple logins. This Framework component should not only support custom applications, is should also support Packaged Applications and Enterprise Services.

**Enterprise Infrastructure Services**

These are services (based on applications) that could potentially be leveraged by the entire user community (external and internal). Most enterprise services are infrastructure components and some of them provide the capability for users to leverage them as an application. Following are some examples of Enterprise Services.

**1.** **Directory Service:** This is the standard directory services provided enterprise wide and generally deployed in conjunction with the eMail service. Most of the enterprise implement meta-directory for managing identity across the enterprise.

**2.** **Personal Information Management:** This is the basically the standard eMail, Calendar, Address book, etc. and includes access to this information from any channel (browser, thick client, mobile devices, etc.).

**3.** **Collaboration:** This provides capabilities such as white board, conference calling, instant messaging, discussion forums, news groups, workspace, etc.

4. **Enterprise Content Management System:** This is the infrastructure service for driving custom applications such as Knowledge Management, Asset / Contract Management, Collaboration, etc. The recommended architecture approach is to leverage the APIs provided by the portal or the content management provider. All the content management system market leaders provide the capability to develop templates for uploading / authoring content as well as workflow for managing approval process.

   Following are the best practices for implementing the enterprise content management system:
   - Define the Taxonomy up front, ideally creating one that is enterprise wide.
   - Create a single document base/repository, enterprise wide. This may not be practical, but is a good goal.
   - Publish all content to one single location in production and configure all applications to

retrieve content from that location (reduces TCO).
- A key success factor is end user training for the authors and the content approvers.
- Partner closely with the content management system provider by engaging their Architects for every project, especially during the design phase of the project.
- Leverage the pre-built portlets to author, review and manage the content.
- Engage a specialized function person from either your SI or Content Management System (CMS) provider to map the business processes to CMS workflow.

5. **Search Service:** Any user (external or internal) should have the ability to find the information they are authorized to access. There are two types of search solutions:
   1. Key Word Search: standard search capability that most users are accustomed to.
   2. Natural Language Search: this is generally targeted towards a non-technical / internet savvy user who has just been introduced to technology and want to find information by asking questions using their local language.

   The integration of a search engines is straight forward. It is generally an XML / HTTP request to the search engine and the engine returns the results in the order requested.

   The search engine goes hand in hand with the content management system. Following are some of the best practices based on our experience.
   - Create one taxonomy enterprise wide for the content management system.
   - Define meta-tags for the content and leverage them in the Portal to present content to the users (based on their entitlements).
   - Use search engines to crawl and create multiple collections / sub-collections as required
   - Leverage federated search between various business units, if required.
   - Leverage portal tags and entitlements to protect secure contents.
   - Recommend storing secure content at the application server level.

   One of the major issues potentially encountered by large sites is the time taken to crawl the entire content repository. There are few alternatives to help resolve this issue such as creating multiple collections and including all the collections in the search criteria, performing partial crawls on a periodic basis, setting up multiple search engines and leveraging federated search, etc. The right strategy to be adopted would be based on the business needs. Our recommendation is to develop the architecture and the process in collaboration with the search solution provider/vendor.

**Enterprise (Role-Based) Portal**
On implementing the Web Tier Components defined in this document, enterprises would achieve the "Current State" as illustrated in the diagram below.

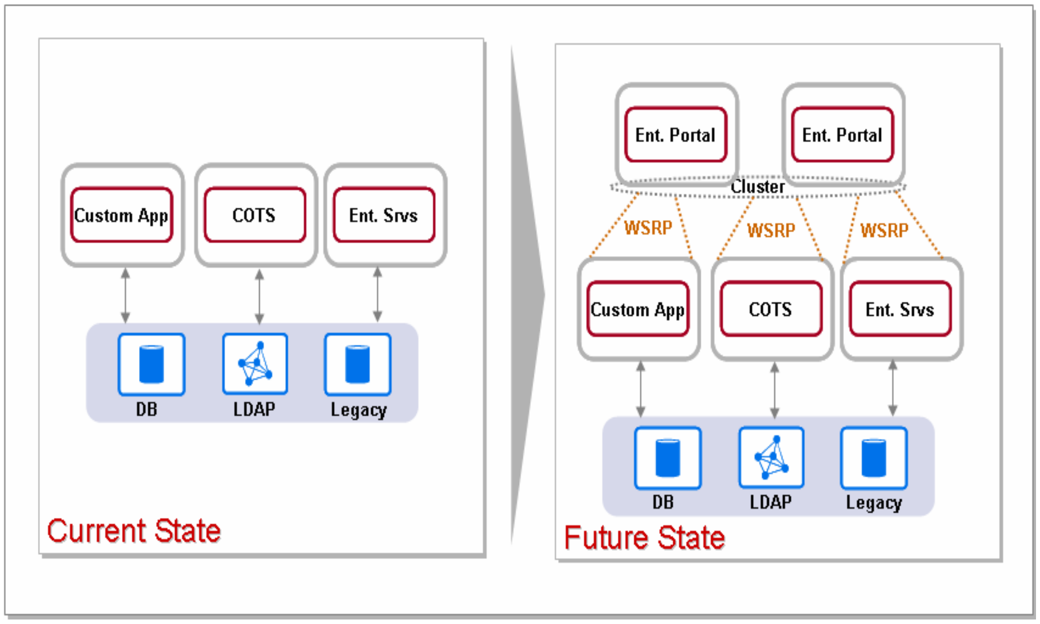


Figure 8 -- Enterprise (Role Based) Portal

In the current state, IT organizations can rapidly deploy business solutions in the form of customer applications, Packaged Applications, Enterprise Services a combination of these components. The Custom Application Framework enables business to provide a great user experience. However, this has the following drawbacks:

- Re-Branding the user experience would potentially require changes to all the sites.
- Users still need to know the URLs for each of the sites. By adopting some best practices, this can be reduced but not eliminated.
- This model results in redundant hardware and software for each of the point-solutions. This is because each of the business units would like to schedule their own maintenance windows and the only way to facilitate this is to have dedicated infrastructure for each of the point solutions.

The Target State is to leverage the concept of Federated Portals to create an enterprise wide Role Based Portal. The advantages of this approach are as follows:

- Single point of entry for all employees, customers, partners, etc.
- Provide Application (Portlets) access based on the role of the user.
- Enable consolidation of infrastructure (both hardware and software).
- Always-ON capability provided by the Enterprise (Role based) Portal.
- Simpler re-branding of sites.
- Multi-Channel delivery provided by the Federated Portal by leveraging Services.

## Service Tier

The Service Tier is the primary enabler of the SOA and includes the components described in this section. It enables integration and business process automation across the enterprise. This tier is based on the SOA principles of coarse-grained, loosely coupled, and standards-based services. It provides the ability to be responsive to changing business needs by providing global solutions, with reduced application and infrastructure complexity, increased reuse of business services and service orchestration capabilities.

### Service Bus

The Service Bus is the key component for delivering the service-oriented infrastructure for IT agility and alignment with business needs. It should have seamless integration with service registry and service management components which in turn accelerates configuration and deployment management and simplifies management of shared services across the enterprise.

The Service Bus should be able to receive any synchronous or asynchronous message in any protocol and route it to the destination based on configuration rules. In addition, it should provide the capability to transform the message to the format required by the destination. As this controls the message flow between the consumer and the producer, the Service Bus is in the unique situation to manage, monitor and enforce the service levels.

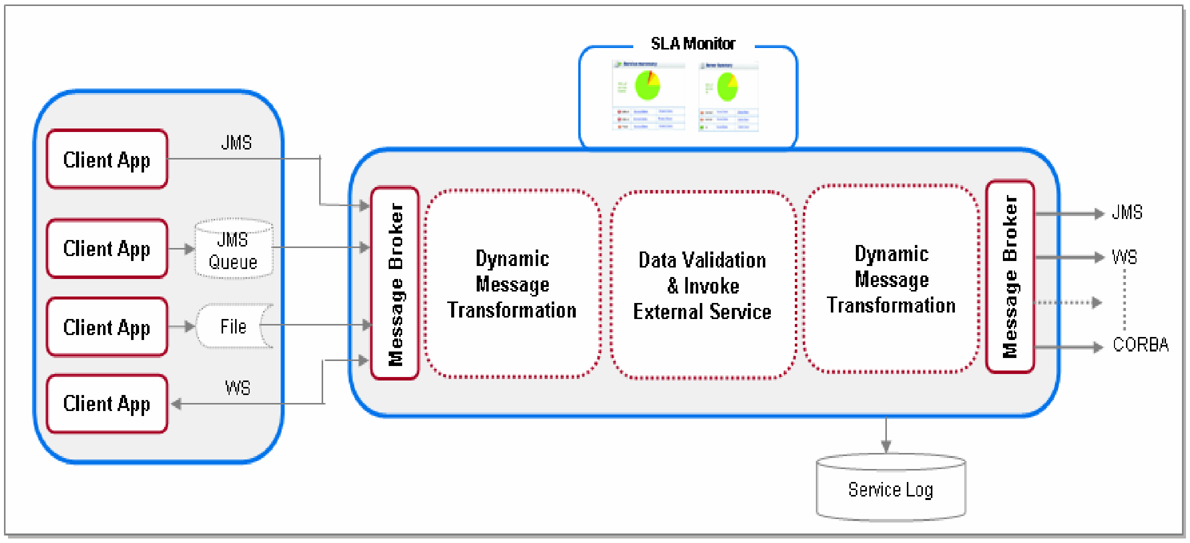


Figure 9 – Enterprise Service Bus Architecture

The above diagram represents the Enterprise Service Bus. The Service Bus basically acts as a dynamic configurable Message and Service broker. Following are the capabilities that define the Service Bus.

- Message Brokering between heterogeneous environments.
    - Support Asynchronous, Synchronous, Publish and Subscribe messaging
    - Support synchronous and asynchronous bridging
    - Support multiple message formats including SOAP, SOAP with attachments, XML, structured non-XML data, raw data, text, email with attachment. etc.
- Support heterogeneous transports between service end points.
    - Support multiple protocols such as File, FTP, HTTP(s), multiple JMS providers, RMI, Web Services, CORBA, DCOM, eMail (POP, SMTP, IMAP), SIP, etc.
- Support message transformation capability which is required to enable the consumer to talk to the producer and is not expected to be leveraged as a full fledged transformation engine.
- Support configuration-driven routing:
    - Message routing based policies or call-outs to external services to support complex routing.
    - Support both point-to-point and one-to-many routing scenarios to support both request-response and publish-subscribe models.
- Provide Monitoring capability:
    - Service monitoring, logging and auditing with search capabilities.
    - Capture key statistics for message and transport attributes include message invocations, errors, performance, volume and SLA violations.
- Provide High-Availability:
    - Support clusters and gather statistics across the cluster to review SLA violations.
- Simplified service provisioning:
    - Deploy new versions of services dynamically through configuration.
    - Migration of configured services and resources between design, staging and production.
    - Support multiple versions of message resources that are incrementally deployed with selective service access via flexible routing.
- Support configurable policy-driven security:
    - Support the latest security standards for authentication, encryption-decryption, and digital signatures.
    - Support SSL for HTTP and JMS transports.
    - Support multiple authentication model.
- Policy-Driven SLA enforcement:
    - Establish SLAs on a variety of attributes including throughput times, processing volumes, success/failure ratios of messages processes, number of errors, security violations, schema validation issues, etc.
    - Initiate automated alerts or operator-initiated responses to rule violations via flexible mechanisms including e-mail notifications, triggered JMS messages, triggered integration processes with a JMS message, Web Services invocations with a JMS message or admin console alerts.

Following are some best practices for the Service Bus.

- It is good practice to start adopting the Service Bus whenever the number of services is more than 50. One definitely needs a service bus when the number of services exceeds 150.
- Start small by targeting a single composite applications or divisional business process that spans multiple systems.
- Multiple LOB could potentially manage their own service bus based on their policies and a Service Bus at an Enterprise level could act as a broker for sharing services across the various business units.
- The functionality described above must be abstracted from the service itself. Organizations must make the decision between deploying a vendor provided Service Bus and internally developed abstraction layer.

**Service Registry**

SOA requires services to be coarse-grained, loosely coupled, and standards-based. As services are developed and deployed there must be a catalog of services available for architects, developers, operations, business, etc.

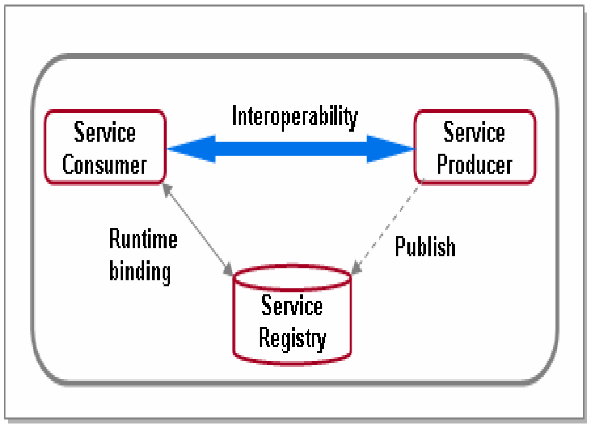


Figure 10 – Service Registry

The above diagram illustrates the architecture of the service registry. The Service Producer publishes the service to the service registry which is leveraged by the service consumer for runtime binding. The registry also acts as the system-of-record for the predefined business policies which could be used at runtime for enforcement of these policies.

Following are the capabilities that should be provided by the Service Registry:

- Core services, including replication, UDDI data store and security.
- Information services, including data validation, SOA mappings, advanced classification, and business data access service.
- Lifecycle services, including approval / change management, change notification, business service discovery and QoS management.
- Configuration web-based business service console.
- Platform-independent open architecture, interfacing with leading enablement, management and security products.

Following are some of the best practices for the Service Registry:

- Similar to the Service Bus - start small and grow over time.
- Every LOB may have its own implementation of the Service Registry and should be replicated to the enterprise service registry.
- Provide service browsing capabilities for architects, developers and operations so as to facilitate re-use and identify service dependencies.
- As this is the system-of-record for all systems, maintain service contract information along with the service definition.
- Version all services.

**Service Manager**

As the SOA implementation matures in an Enterprise, there is a need for an overall service manager. The primary function of this service manager is to manage, monitor and report on all the services enterprise wide. Following are some of the capabilities that Service manager needs to provide:

- Manage and ensure that the Service Level is maintained enterprise wide
- Map and maintain service hierarchy across the enterprise and provide dependency matrix to operations.
- Detect and manage exception conditions.
- Review and monitor business transactions, provide capability to review in-flight transactions.
- Manage service lifecycle and validate before deployment.
- Provide non-intrusive service discovery across multiple systems.
- Ability to manage and integrate with multiple Service Bus and Service Registry infrastructures.
- Integrate with existing Monitoring infrastructure.

**Shared Data Service**

For this version of the reference architecture we shall focus only on the Enterprise Information Integration EII capability. EII refers to software systems that can take data from a variety of internal and external sources and in different formats and treat them as a single data source.

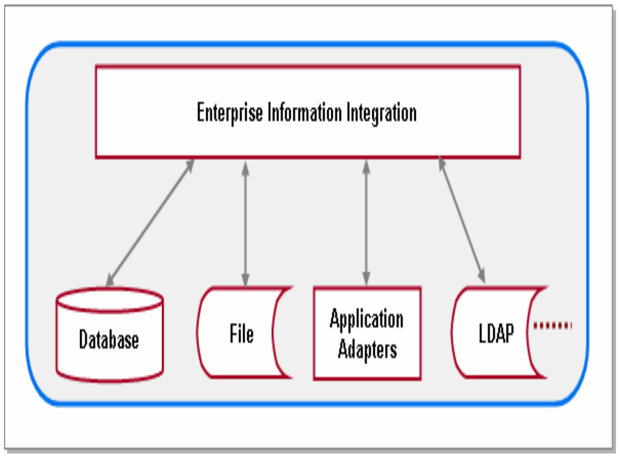


Figure 11 – Enterprise Information Integration (EII)

Following are the capabilities that should be provided by EII:

- Provide data modeling capability across multiple sources.
- Develop query (read and write) to extract information from multiple data sources.
- Support multiple data sources – Database, File, Application Adapter, LDAP, Web Services, etc.
- Provide data transformation Capability.
- Provide data validation capability.
- Expose data services to client applications – RMI or Web Services.
- Standards based – SQL, XQuery, XML, Web Services, JDBC, J2EE, etc.

***Note***: Even thought the Service Data Object (SDO) standards have been defined to simplify and unify the way in which applications handle data, the industry has not yet clearly defined the standards for EII. Each vendor has their own extensions that deal with reading, updating and inserting data to each of the data stores. We would collectively prefer to see the vendors, through the standards bodies, agree on one consistent approach.

**Business Process Management**
The BPM is used to manage long-running business processes, both synchronous and Asynchronous. Light weight service orchestration is usually performed by the Service Bus. Following are the function / features that should be provided by the BPM.

- Visual Process Modeling – modify views, business process models, etc.
- Built on open standards – BPEL a plus.
- Business process orchestration and automation between private processes, public processes, human tasks, error handling as well as supporting nested and concurrent processes for advanced modeling and custom logic as required to enable rapid customization.
- Optimized process performance: Allows flexibility of configuration for state-full (long-running) and stateless (short-running) process design patterns as well as synchronous and asynchronous process execution.
- Status monitoring: Allow the user to monitor status of end-to-end processes graphically and measure performance vs. service level agreements.
- Process instance monitoring: View statistics on running processes; drill into individual details; terminate, delete, or suspend problematic process instances.
- Enable end users—task creators, task workers, and task administrators—to interact with running business processes for handling process exceptions, approvals, status tracking, etc.
- User and Group Management: Centralize the assigning of roles, users, and groups working on integration projects.
- B2B Protocol Support: Enable rapid, secure online connection with suppliers and customers via leading standard protocols such as RosettaNet, ebXML, and EDI, with secure messaging, digital signatures and encryption, recoverable and trackable messages, and dynamic configuration updating.

**SOA Frameworks**
SOA Frameworks are re-usable web services that can scale and be consumed by a wide variety of applications form the backbone of the SOA. These re-usable services must be enterprise-class and designed well enough to scale under load, and meet demands of a diverse audience of stakeholders.

SOA frameworks catalyze and support the move to an SOA by helping development teams to rapidly design, develop and deploy well-designed, modular, flexible, scalable and supportable web services, web applications and portlets. As companies start adopting SOA principles to transform their IT architecture, it will be very important for the underlying services to be created in a consistent, repeatable manner with enterprise qualities in mind

A framework can be defined as a reusable, “semi-complete” (skeleton) application designed to be extended to build specific services or applications. Frameworks improve and enable consistency in the delivered software. Frameworks invert the control between themselves and the application or services that are created on top of them.

Frameworks typically provide a set of higher level programming abstractions and a vastly superior starting point for creating enterprise class services. Frameworks also very often specify a layered architecture for services that incorporates several design patterns and software engineering best practices. The architecture specifies the responsibilities of the components in each of the layers and the collaboration between them.

Services based on the frameworks inherit the good architecture and best practices that have been incorporated into the frameworks themselves. This makes it possible to ensure that a team of average developers is able to develop well architected services that take advantage of design patterns and best practices.

The typical layers that a services creation framework would offer include:

- *Transformation Layer*: Supports protocol and data-type conversions to support multiple access protocols, while at the same time keeping most, if not all, of the service implementation protocol and access mechanism agnostic.

- *Business Logic Layer:* Holds all the business logic in the system. This includes such abstractions as Request, Result, UseCaseController, BusinessPolicy objects etc.

- *Business Data Layer:* The layer for domain objects, i.e. the objects that have a consistent definition across many applications in the enterprise. The Business Data Layer should provide location transparency - that is, the users of the domain objects should not be concerned about the exact physical location of the underlying persistent data on which the domain object itself is based. This layer should be able to manage persistence to, and retrieval from a multitude of persistence repositories in the enterprise.

- *Integration Layer:* A placeholder for a myriad of connection technology implementations ranging from JDBC, to JNI to Java Connectors. All the infrastructure code that is needed to access extended enterprise systems such as ERP systems, content repositories etc. will fit into this layer.

SOA Frameworks offer a number of benefits for both developers and the corporation. For developers, the frameworks offer the following benefits:

- A solid foundation to create services, web applications and portlets.
- Improved productivity as a result of using a framework that incorporates design patterns and best practices.
- Utilize off-the-shelf features of the frameworks and write less code.
- Don't need to understand the nuts and bolts of J2EE standards and specifications.
- Don't need to be an expert at Object-Oriented design and design patterns to benefit from using them.

For IT organizations and the company as a whole, the SOA Frameworks offer the following benefits:

- A catalyst for getting to a Services Oriented Architecture quickly and at a lower cost.
- Consistency of design and development across projects.
- Repeatability and the ability to guarantee a minimal level of architecture and design rigor.
- Improved business agility as a result of having modular solutions that can be changed easily (often via configuration changes).
- Use of software engineering best practices amongst developers with varying skill levels.
- More consistent, predictable and better tested solutions.
- Improved mobility of developers to move from one project to another.

IT organizations are using SOA principles to aggressively create re-usable services that encapsulate and expose key business processes. By combining a layered architecture, ease of use and a deep emphasis on good architecture and re-use, SOA frameworks enable the creation of enterprise-class mission-critical services in a vendor neutral, portable manner.

## Application Tier

This is the Tier where IT organizations have made the largest investment. Even though enterprises will invest in SOA moving forward, this tier is not going to go away anytime soon.

### Legacy Applications

These are the thick client applications still in existence within an enterprise. They may the old versions of packaged applications that have not yet been upgraded due to budget / business constraints. Alternately, these applications are best suited to run in the client / server mode.

These applications will typically have some published APIs or Logical Models for integration purposes.

### Mainframe Applications

Business solution provided by proprietary mainframe systems and integrations to these applications are either through a messaging or database gateways. There is an effort underway by the mainframe software vendors to expose all their APIs as Web Services, which in most cases is not the right approach. In most cases, the best approach is to leverage a middleware to develop an abstraction layer and expose the services at the right level as required to support the business requirements.

### Enterprise Application Integration

This is the traditional approach to enterprise application integrations. EAI middleware software typically provides the following capabilities:

- Messaging: Message Oriented Middleware (MOM) with ability for resources to publish and subscribe to messages.
- Business Process Manager: proprietary BPM capability to automate business processes.
- Application Adaptors: Pre-built Connectors to various packaged applications that allow access to the application views or technology adaptors to other technologies like databases, messaging (MQ, JMS), Web Services, etc.

The best practice for EAI is to leverage an integration Hub. However this by itself, without the appropriate supporting methodology, may result in a point-to-point solution on the hub. Even though the Services Tier provides the Service Bus and BPM, which enables enterprises to move adopt SOA and migrate away from EAI, this migration is expected to take a long time. Especially as large IT organizations have invested very heavily in EAI and replacing this capability will take a while.

### Enterprise Security

Currently most of the applications, whether they are packaged or custom applications, implement their own security solution. Most of the applications provide the ability to externalize their authentication but rely on an external implementation which results in redundant code. In addition, the administrative cost of duplicate user accounts on multiple applications can be huge.

This could be simplified by breaking the enterprise security into three major components:

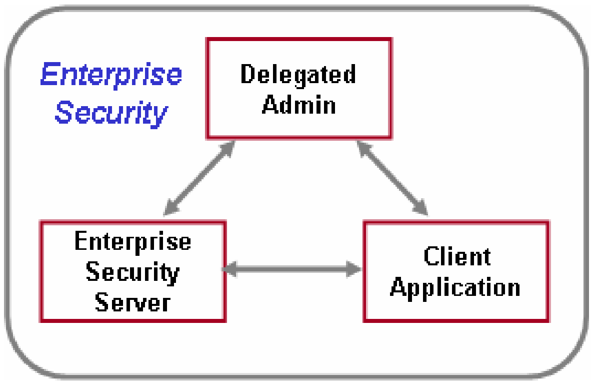


Figure 12 – Enterprise Security Components

*Delegated Admin:* User and Resource Administration application that enables administrators (based on their role) create/modify/delete user privileges. This application updates the same repository leveraged by the Enterprise Security Server.

*Enterprise Security Server:* Provide security services such as User Authentication, User Identity Management, Authorization, Auditing, User Profile Management and User provisioning.

- Identity Management involves managing user identities within and across multiple applications of an enterprise. Mapping of multiple identities to a single user or linking a user identity in one application with a different identity of the same user in another application allows multiple legacy user identities to co-exist.
- Authentication involves validating the identity of a user. Several authentication mechanisms may be used in an enterprise with the most common one being validating against a password. Other mechanisms may involve digital certificates, smart cards etc. Enterprise-wide policies can be instituted to ensure that users present conclusive proof of identity before being provided access to resources. From a convenience and usability perspective, users may need to be able to sign on once and gain access to multiple resources. However, this also increases the need to add proper controls and policies to ensure that once authenticated a user doesn't gain access to resources that he/she should not have access to.
- Authorization involves controlling access of users to diverse resources across the enterprise.
- Auditing involves tracking user activities.
- Profile Management involves managing the profiles of the users.
- Provisioning automates the tedious and time-consuming process of managing accounts and their life cycle. It allows centralized activation, modification or deactivation of accounts across multiple applications in an enterprise. In an enterprise, user provisioning could include explicit granting or revoking of user access to resources and establishing of entitlement policies for the user.

*Clint Applications:* The client applications externalize and leverage the enterprise security services.

## Additional Information

### Acknowledgements

The authors would like to acknowledge the many organizations and individuals that contributed portions of this document, performed substantial editing of the content, or who provided reviews and feedback. Specifically, we would like to acknowledge:

**Erik Dahl,** SVP Lead Integration Architect, Bank of America
**Robert Eisenberg**, Principal, REA Associates
**Ashok Kumar,** Manager, SOA Architecture, Car Rental Group
**Jeffery Lamb,** Enterprise Architect, Wells Fargo
**Tom Mitchell,** Lead Technical Architect, Wells Fargo – Private Client Services
**Burc Oral,** Principal, Dev Atma Technologies, Inc.
**Yogish Pai**, Chief Architect, AuqaLogic Composer & Chair – SOA Blueprint working groups
**John Schmidt,** SVP Architecture/Engineering, Bank of America
**Sankar Ram Sundaresan**, Chief Architect, e-Business IT, Hewelett-Packard Company

### Copyright