Oracle® Reference Architecture

SOA Infrastructure

Release 3.0

E14479-03

September 2010

ORACLE®

ORA SOA Infrastructure, Release 3.0

E14479-03

Primary Author: Anbu Krishnaswamy

Contributing Author: Stephen Bennett, Dave Chappelle, Bob Hensle, Mark Wilkins, Jeff McDaniel, Cliff Booth

Contributor: Sazi Temel, Martin Cookson

Contents

List of Figures

Send Us Your Comments

ORA SOA Infrastructure, Release 3.0

E14479-03

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this document?

If you find any errors or have any other suggestions for improvement, please indicate the title and part number of the documentation and the chapter, section, and page number (if available). You can send comments to us at its_feedback_ww@oracle.com.

Preface

Service Oriented Architecture is an IT strategy which spans the entire enterprise. IT projects are implemented to fit within an SOA and are therefore just a piece of the larger SOA initiative. SOA Infrastructure projects differ slightly from the typical SOA project delivery, in that the focus is mostly on enabling infrastructure products, rather than developing new business function, or providing for other business driven needs. The focus of SOA Infrastructure is to enable the delivery teams to deliver SOA projects faster, as well as make the overall SOA undertaking much more manageable. There are two drivers for realizing SOA Infrastructure, as this technology plays a role in both design time and runtime activities.

The EA teams are responsible for establishing the high level enterprise SOA reference architecture, and planning the procedures and guidelines that need to be followed when developing applications and services that will be considered participants of the SOA. These planning and strategizing activities place demands on SOA Infrastructure to make these aspects of SOA possible without having to develop infrastructure in house. Design time demands include application and service composition, service orchestration, loose coupling, controlled service discovery, service versioning, managing security policies and data format conversions.

From the runtime side of things, it is crucial for SOA success that the operations team fully control and maintain the operational environment as the SOA matures over time. These teams need infrastructure to assist with monitoring and managing the SOA environment.

These demands surface from the concepts brought to bear by the adoption of SOA and the generally accepted practices of how an enterprise goes about adopting SOA. Common infrastructure products, which are categorized as SOA Infrastructure, include service bus, service repository and registry, service management and service oriented security products; are utilized to make SOA possible.

Document Purpose

This document enumerates the key capabilities required for SOA implementations and organizes them into logical architectural components. Oracle Fusion Middleware products are mapped to the logical architecture and various views of the SOA Infrastructure including physical, deployment, process and development views are elaborated in detail. This document provides an understanding of the best way to implement an effective SOA infrastructure.

Audience

This document is primarily intended for Infrastructure Architects responsible for building SOA infrastructure. Enterprise Architects and Project Architects that want to understand the best way to build applications, processes, and Services will gather valuable insight from a good understanding of the capabilities of the SOA infrastructure. Developers will also gain sufficient understanding of the various SOA development infrastructure components required to build Services and composite applications.

Document Structure

This document is organized into the following sections.

Chapter 1 - gives an introduction to SOA infrastructure.

Chapter 2 - defines the key infrastructure capabilities required for SOA implementations.

Chapter 3 - defines the logical architecture and maps the capabilities to the core logical components

Chapter 4 - defines the product mapping view of the SOA infrastructure which describes and maps the Oracle Fusion Middleware products to the logical architecture.

Chapter 5 - describes the deployment view of the SOA infrastructure.

Chapter 6 - explains the process view and describes the runtime infrastructure process and SOA management process in detail.

Chapter 7 - covers the development view of the SOA infrastructure and goes over the design time process and design time tools.

Chapter 8 - provides a summary of the SOA Infrastructure document.

Appendix A - provides a list of documents and URLs for further reading

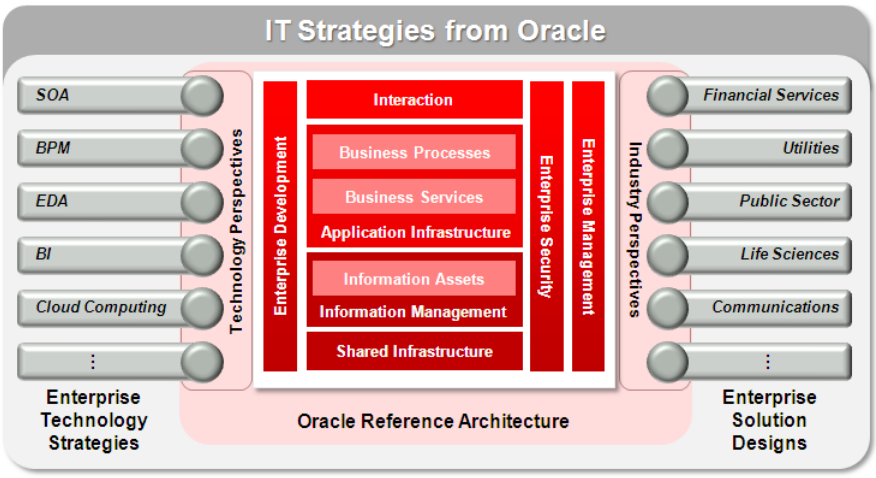
How to Use This Document

This document should be read by everyone that is interested in learning about architecting and building an enterprise class SOA infrastructure. It is one of the documents in the collection that comprise Oracle Reference Architecture.

This document can be read from beginning to end or as a reference. If specific infrastructure components are not applicable to you at this point of time, you can skip that part but be sure to read the interdependencies of the technologies/products to ensure that there are no holes in the architecture.

Related Documents

IT Strategies from Oracle (ITSO) is a series of documentation and supporting collateral designed to enable organizations to develop an architecture-centric approach to enterprise-class IT initiatives. ITSO presents successful technology strategies and solution designs by defining universally adopted architecture concepts, principles, guidelines, standards, and patterns.



ITSO is made up of three primary elements:

- **Oracle Reference Architecture (ORA)** defines a detailed and consistent architecture for developing and integrating solutions based on Oracle technologies. The reference architecture offers architecture principles and guidance based on recommendations from technical experts across Oracle. It covers a broad spectrum of concerns pertaining to technology architecture, including middleware, database, hardware, processes, and services.
- **Enterprise Technology Strategies (ETS)** offer valuable guidance on the adoption of horizontal technologies for the enterprise. They explain how to successfully execute on a strategy by addressing concerns pertaining to architecture, technology, engineering, strategy, and governance. An organization can use this material to measure their maturity, develop their strategy, and achieve greater levels of success and adoption. In addition, each ETS extends the Oracle Reference Architecture by adding the unique capabilities and components provided by that particular technology. It offers a horizontal technology-based perspective of ORA.
- **Enterprise Solution Designs (ESD)** are industry specific solution perspectives based on ORA. They define the high level business processes and functions, and the software capabilities in an underlying technology infrastructure that are required to build enterprise-wide industry solutions. ESDs also map the relevant application and technology products against solutions to illustrate how capabilities in Oracle's complete integrated stack can best meet the business, technical, and quality of service requirements within a particular industry.

ORA SOA Infrastructure, along with *ORA SOA Foundation*, extend the Oracle Reference Architecture. They are part of a series of documents that comprise the SOA Enterprise Technology Strategy, which is included in the IT Strategies from Oracle collection.

Please consult the ITSO web site for a complete listing of SOA and ORA documents as well as other materials in the ITSO series.

Suggested Pre-reading

The following documents are suggested pre-reading for those that would like to more fully understand the concepts this document builds upon.

Service Engineering - An Overview document lays out the process and techniques needed to develop enterprise class SOA Services. It provides an approach to augment

the traditional solution delivery methods with the changes and additions required for SOA Service development and mangement.

ORA SOA Foundation document describes the concepts of SOA and defines the Service Layers and SOA Infrastructure layers that are referred in this document.

Conventions

The following typeface conventions are used in this document:

Convention	Meaning
boldface text	Boldface type in text indicates a term defined in the text, the ***ORA Master Glossary***, or in both locations.
italic text	Italics type in text indicates the name of a document or external reference.
underline text	Underline text indicates a hypertext link.

In addition, the following conventions are used throughout the SOA documentation:

"Service" v. "service" - In order to distinguish the "Service" of Service Oriented Architecture, referred to throughout the SOA ETS document series, the word appears with its initial letter capitalized ("Service"), while all other uses of the word appear in all lower-case (e.g. "telephone service"); exceptions to this rule arise only when the word "service" is part of a name, such as, "Java Message Service" ("JMS"), "Web Service", etc.

1

Introduction to SOA Infrastructure

Infrastructure as it relates to a Service Oriented Architecture (SOA) is the basic technologies and building blocks necessary to make a SOA work effectively in an enterprise. SOA is much larger than applying technology and has organizational and procedural aspects as well. This document will focus on the physical tools and technologies that implement the SOA infrastructure.

1.1 SOA Infrastructure is different

The key difference between SOA Infrastructure and other types of infrastructure is summarized below:

- SOA infrastructure is based on standards allowing you to choose from various products and vendors which are best suited to meet your requirements, yet still interoperate with other service based technology which you may have previously adopted.
- Highly distributed, heterogeneous nature of SOA attempts to bring a number of disparate moving parts together, making it very complex naturally. This very characteristic of SOA infrastructure calls for due diligence and careful planning when designing and building the SOA infrastructure.
- Services are more granular than applications. So the infrastructure should be able to support the distribution, deployment, discovery and management of these granular artifacts as opposed to the monolithic applications, which are much easier to deploy and manage.
- Typically application infrastructure provides the initial foundation for deploying applications but beyond that point only upgrades to the infrastructure and applications would be needed. In contrast, SOA Infrastructure is offered in multiple independent products that can be deployed in different order based on the specific requirements and strategy of the enterprise. This allows the purchase of infrastructure needed to continue the path of maturing SOA independently.

The concepts of Service Oriented Architecture have been around for some time now; however, until recently, there has been little standards-based infrastructure technology to enable SOA and make it realistic to achieve. The pathway to SOA Infrastructure that has emerged today is a logical evolution built on past infrastructure and largely based on standards.

Some of the major challenges that arise with SOA without the assistance of a SOA infrastructure include:

- Avoiding tightly coupled service connectivity which results in a rigid and inflexible architecture

- Achieving a consistent data format and representation of enterprise data entities
- Understanding which services are available, where they reside, their contract, invocation protocols, and rules for use.
- Monitoring and enforcing quality of service such as service level agreements described by service contracts.
- Managing service versioning and service life cycle requirements.
- Establishing the centralized management of security policies for SOA participants.
- Achieving flexibility where coarse grained services and applications can be composed of existing services.
- Invoking services over heterogeneous transports using varying message brokering capabilities.
- Achieving the performance and SLA requirements in a highly distributed and heterogeneous environment.

The focus of SOA infrastructure is to provide the foundational components and capabilities required to address challenges such as those listed above. This document provides the architectural guidelines for applying SOA infrastructure to address these challenges.

The approach of this document is to begin with the SOA conceptual reference architecture that was discussed in the *ORA SOA Foundation* document, present the infrastructure capabilities that are required for a successful SOA deployment and define the logical components. The Oracle Fusion middleware SOA products are then mapped to the logical architecture to demonstrate how the SOA infrastructure can be realized using Oracle technologies.

1.2 SOA Infrastructure Principles

This section defines some of the principles of SOA infrastructure. These principles provide the guidance for creating a sound SOA infrastructure.

- **Standards Support**
  - The SOA infrastructure must be based on open standards. This supports a best of breed approach and prevents vendor lock in.
  - SOA Infrastructure must support all services with standards-based interfaces, regardless of their choice of implementation technology.
  - Choice of service hardware, operating system, or implementation language should not constrain choices for service consumer.
  - The SOA infrastructure must support reliable messaging (e.g. WS Reliable Messaging, JMS, etc.)
- **Data Management**
  - The exchange of data between internal and external systems must be logged and time-stamped to the extent necessary to maintain consistent documentation and transaction trace of this communication. The SOA Infrastructure will provide communication logging and audit trail capabilities, but logging on business level must be maintained by each service implementation.
  - Data replication must be avoided. If an application demands a local representation of data in order to operate, control on the integrity of the

replicated data is the assumed responsibility of the demanding application and not the responsibility of the SOA Infrastructure.

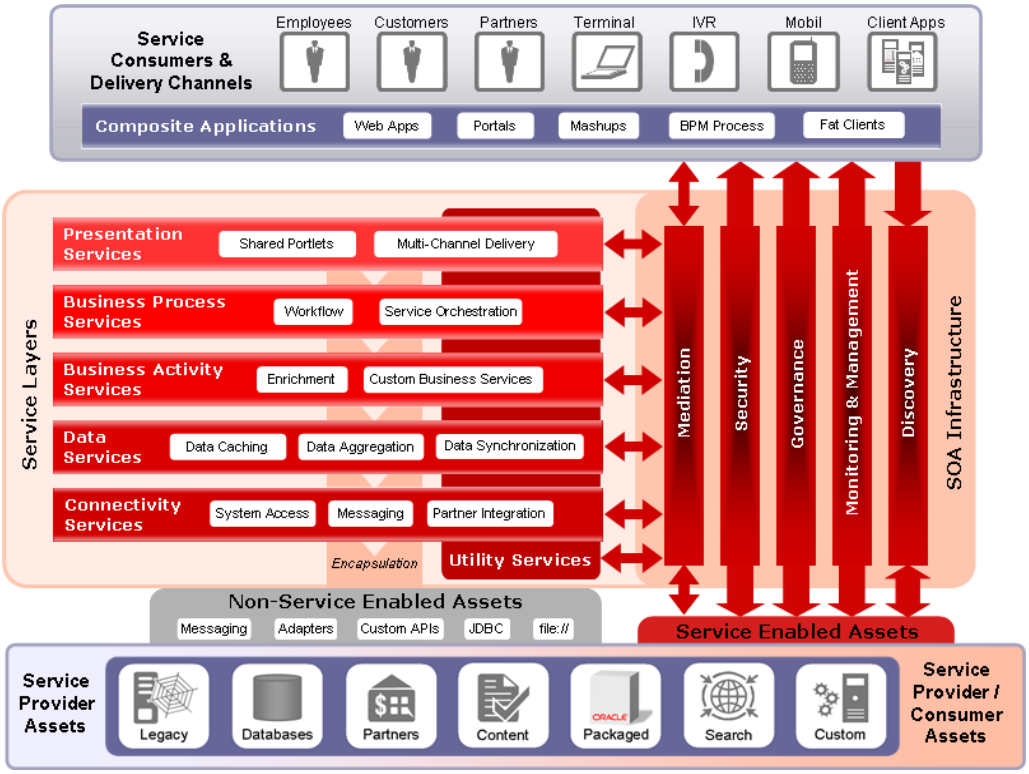
- **Infrastructure Capabilities**
  - The SOA infrastructure must provide the capabilities to deploy, publish, discover, invoke, monitor and manage services.
  - Services must utilize the connection management and recovery functions provided by the SOA infrastructure.
  - The scaling of the SOA infrastructure must be based on the requirements from the business. The technical infrastructure must be scalable to support the requirements on involved parties regarding performance, response time, availability, network connections and capacity.
  - SOA infrastructure must support multiple versions of the service concurrently and have ability to introduce new versions of the service without requiring all the consumers to change simultaneously.
  - SOA infrastructure must provide location transparency capabilities to decouple consumers from providers.
  - All integration between disparate business components must take place through the middleware provided as part of the SOA Infrastructure.

2

SOA Infrastructure Conceptual View

A good place to start the SOA infrastructure discussion is with the conceptual view of the SOA Reference Architecture. This has been discussed in the *ORA SOA Foundation* document in detail. Figure 2–1 illustrates the common high level capabilities of a SOA:

Figure 2–1 SOA Conceptual Architecture



The center of the diagram depicts the conceptual elements of an SOA. At a high level, it is comprised of Services and SOA infrastructure. The infrastructure helps provide features to promote qualities such as **mediation**, security, governance, monitoring, management, and discovery. The Services layers are covered in the *ORA SOA Foundation* document. This document focuses on the SOA Infrastructure shown in the diagram. The SOA infrastructure provides the common capabilities for development, deployment and integration of the Services.

A key benefit to SOA, and most likely reason to succeed, is that many companies have worked together to define standards for interoperability. Though there are many standards, versions, and options, the result of this work is still beneficial. While

previous attempts to promote interoperability fell short due to proprietary extensions and implementations, SOA has a much better foundation to work from in this regard than perhaps any previous wave of technology. For this reason, standards support should be considered a priority when defining the SOA reference architecture and the infrastructure therein.

Several industry standards have been defined around SOA and SOA infrastructure technologies. These standards have been covered in the *ORA SOA Foundation* document. Given the unique nature of the SOA infrastructure and the ability to create a best of the breed SOA infrastructure using disparate technologies, it is imperative to take a standards based approach in building the infrastructure. When an organization chooses the tools and technologies for its SOA implementation, it needs to consider the extent of standards support in these technologies and ensure that it will meet the requirements of that organization.

2.1 SOA Infrastructure Capabilities

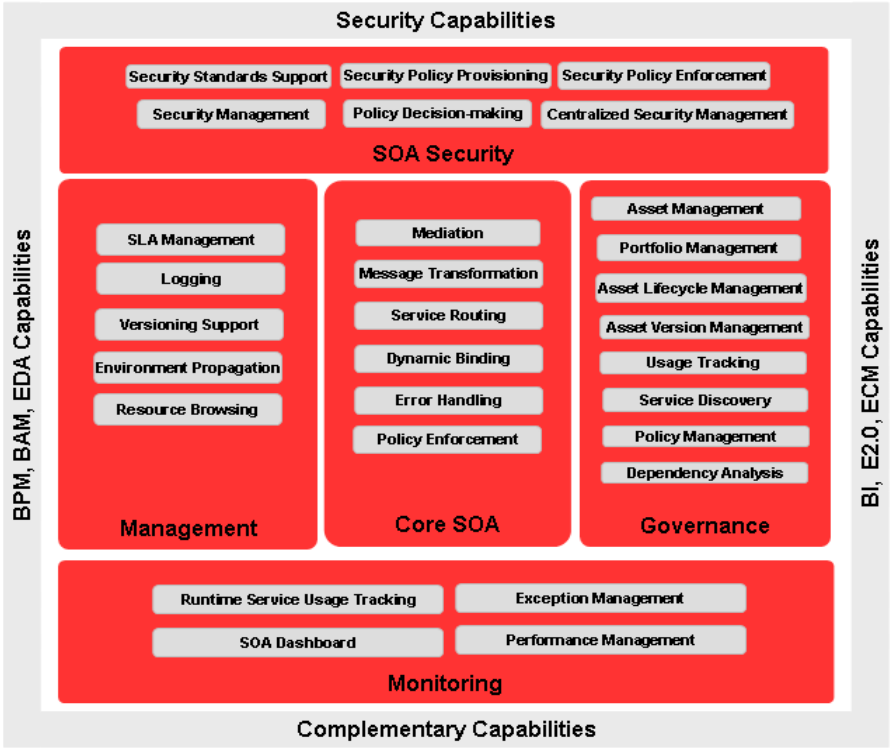
Much the same way that container-based infrastructure provides core services to J2EE based applications, SOA infrastructure provides the enabling capabilities required to realize and mature a SOA deployment. SOA infrastructure provides the technology that is a common need among all SOAs and allows enterprises to focus on enabling business capabilities within their SOA, rather than building enabling technology.

Though all SOAs can benefit from SOA infrastructure, and all may need the same capabilities (to some extent), the type of products and technologies chosen and the order in which they are deployed may vary from one organization to another. Some products may overlap in features with others, which can lead to confusion over which type of product to use in a particular situation. It is important to decide on the role of each infrastructure component, and the technologies to embrace, before selecting products. These decisions are best captured in a reference architecture document, which can then be shared with architects and development teams across the organization. The intent is to establish a consistent approach, which helps promote interoperability and reuse.

An effective deployment and adoption of SOA will require some key infrastructure capabilities. The infrastructure capabilities must be aligned with the business goals to enable quicker delivery of business capabilities.. This will enable the business to achieve faster Time-To-Market benefits of SOA. A prudent SOA investment will require that the infrastructure capabilities are built in a prioritized order. Growing the infrastructure capabilities is a prerequisite to improving SOA maturity as discussed in a later section.

Figure 2–2 organizes the key SOA capabilities into five primary domains - Core SOA, Management, Governance, SOA Security and Monitoring.

Figure 2–2 SOA Infrastructure Capabilities



Some SOA infrastructure capabilities are dependent on basic infrastructure capabilities like messaging and security. There are also technological strategies like BPM and EDA that complement SOA. The capabilities of these technological infrastructures can leverage and build upon the SOA infrastructure capabilities. This document does not cover these in detail but they will be addressed in the appropriate technology perspective documents. A brief overview of these capabilities is listed in Section 2.7.

2.2 Core SOA Infrastructure Capabilities

This section describes the core SOA infrastructure capabilities essential for building an enterprise-class SOA. The core SOA capabilities include mediation, message transformation, service routing, dynamic binding, error handling and policy enforcement.

2.2.1 Mediation

Mediation can be broadly defined as resolving the differences between two or more systems in order to integrate them seamlessly. A typical IT architecture has a variety of systems and components that are fundamentally different. A better alternative to embedding the mediation logic into each of these systems would be to provide the mediation capability in the SOA infrastructure.

A Service Oriented Architecture can exist with services exposed using alternate means in addition to web services. In order to be flexible and allow for a wide variety of heterogeneous service invocation techniques, SOA will need the support for transport mediation, multiple message formats, over various invocation strategies. Transport mediation bridges the differences in the communication protocols like HTTP, File and FTP.

2.2.1.1 Message Exchange Pattern (MEP) Mediation

MEP mediation resolves the difference between the invocation patterns of the consumer and provider. The most common MEP used in SOA is the synchronous request/response pattern. Asynchronous one way fire and forget, Asynchronous request/response and Robust one way are some of the other popular MEPs that should be supported by the SOA infrastructure. A good SOA infrastructure should also be able to mediate different MEPs using Sync-to-Async or Async-to-Sync bridging.

2.2.1.2 Transport Mediation

Transport mediation allows the consumer to invoke a service using a different transport than that supported by the provider. The SOA infrastructure achieves it by translating the transport protocol and transport headers. The set of supported transport protocols is important to consider when selecting SOA Infrastructure for composition. Some of the common routing transports that may typically be used to invoke services or pass messages within an SOA are HTTP(S), File, FTP, JMS , RMI, IIOP, **JCA**, and POP/SMTP/IMAP.

Multiple transport protocols may be used in order to address various Quality of Service (QoS) requirements. While SOAP over HTTP(S) is the most commonly mentioned form of transport due to its ubiquitous nature, it does not often support reliable messaging and transactions. Even though WS-ReliableMessaging and WS-Transactions specifications have been written, not all entities support them. Therefore it is quite common to leverage a transport that does, when such qualities are needed.

2.2.1.3 Security Mediation

Just as not all service providers support the same technologies and transports, they do not all support the same security implementations. Therefore in order to be able to mediate between endpoints, the infrastructure must be able to mediate issues of security. That is, it must be able to meet the security requirements of a service by extracting and converting security information provided to it by the service consumer. In this way it is acting as a converter rather than a negotiator between endpoints. More details on the security mediation can be found in the *ORA Security* document.

2.2.2 Message Transformation

Transformation is a type of mediation that is a critical capability of any SOA infrastructure. The ability to manipulate and transform messages as they travel from consumer to producer and optionally back to the consumer provides the designer with a great deal of flexibility. These capabilities provide support for such concepts as transformations to and from canonical message formats. They also allow for services to be utilized for a wider audience, by being able to adapt the input and output to a service invocation. Message transformation might include aggregation, enrichment, filtering and wrapping.

2.2.3 Service Routing

Routing refers to the ability to control where a message is sent or which service endpoint is invoked. This can be accomplished by extracting data that affects routing decisions from the contents of the request message (content-based routing), message header or via configuration data (config-based routing).

Loose coupling is the ability to decouple the service consumer from the implementation specifics of the service provider. With loose coupling a service

provider can be versioned, physically moved, or replaced without affecting the service consumer. Routing capability enables the loose coupling required for SOA. These features are fundamental to managing and maintaining a mature SOA deployment.

2.2.4 Dynamic Binding

Dynamic binding refers to the act of connecting a service consumer to a producer at runtime. The SOA infrastructure must determine which service producer to use and return endpoint information to the consumer, who then accesses the appropriate endpoint.

Dynamic binding satisfies the need for loose coupling; however, it requires two round trip interactions, at least the first time - one between the consumer and SOA infrastructure, and the other between the consumer and producer.

2.2.5 Error Handling

Error handling capabilities allow expected and unexpected error conditions to be planned for and handled appropriately at design time. Error handlers are placed within a particular scope which allows errors to be isolated and handled at the most appropriate point; thereby providing the highest likelihood of successful recovery from the error.

2.2.6 Policy Enforcement

An important capability of the SOA infrastructure is the ability to define and enforce policies. By nature, SOA infrastructure is loosely coupled and distributed. Policy enforcement provides a way to define common policies independent of the service implementation and apply them on an as needed basis. Policy enforcement can happen at different enforcement points of the infrastructure based on the use case. Policies can also change independent of the Services or applications allowing a higher degree of agility and control. Policies may include regulatory compliance rules, business policies, security policies, SLA policies and validation rules. The SOA infrastructure should be able to support the enforcement of a variety of policy types.

2.3 Governance

An effective SOA Governance will require a minimum of the following capabilities.

- Asset Management
- Portfolio Management
- Asset Lifecycle Management
- Asset Version Management
- Usage Tracking
- Service Discovery
- Policy Management
- Dependency Analysis

Each of these SOA governance related capabilities is described below:

2.3.1 Asset Management

Asset management is a key governance capability of a SOA infrastructure. SOA requires the management of the metadata of a variety of assets like service artifacts, contracts, policies, and schemas. A robust asset management infrastructure that can maintain the metadata of these assets and relationships between them is a critical aspect of effective SOA governance.

2.3.2 Service Portfolio Management

An important point to recognize with SOA is that the creation and lifecycle maintenance of Services adds some amount of overhead to the development process. The amount of overhead varies, depending on the formality, complexity, and rigor involved in identifying, discovering, creating, managing, and governing services. Even if steps are taken to greatly reduce overhead, one can expect some amount to exist.

Given this acknowledgement of overhead, services should be planned and managed to best satisfy the needs of business and IT. This requires the ability to know what Services exist, what capabilities they provide, and ideally, what business processes they pertain to. The service portfolio is a way to represent this knowledge, communicate it, and plan for future Services.

2.3.3 Asset Lifecycle Management

A corollary to portfolio management is lifecycle management. Services have a lifecycle, starting with project requirements, continuing with identification and creation of a service, deployment, and ending with eventual retirement. Lifecycle management involves the strategy and planning around stages of the service lifecycle. It supports governance in terms of what services are approved and the status they are given. It also supports release planning, as lifecycle management includes control over service releases on which applications and other services may depend on.

2.3.4 Asset Version Management

A natural extension of lifecycle management is version management. Services may require changes based on a number of factors, such as code defects, functional changes, non-functional changes, resource changes, etc.

Versioning can be fairly simple for services with only one consumer. In some cases the consumer and provider may be updated simultaneously. This reduces (or eliminates) the need for concurrent versions.

Versioning becomes more difficult when services are used by multiple consumers. In these cases, every effort should be taken to support multiple concurrent versions in production. This allows each consumer to test and migrate to the newer version under their own release cycles.

2.3.5 Usage Tracking

Usage tracking comes in three forms:

- the registered intent to use a service
- the interest in a service or service asset resulting in a subscription to that asset
- the actual audit trail of consumers invoking services.

The first two are design-time tracking mechanisms, while the third is a run-time approach. All are important in their own way.

There are certain benefits that can be realized through the use of **Usage Agreement**s, aka **Consumer Contract**s. These documents formalize the intent to use a service, which removes ambiguity in terms of who is relying on a service and what performance, load, and availability requirements exist. Consumers may feel more comfortable under such an agreement because their needs are well known. Providers can use this information for lifecycle management and capacity planning. These benefits make it a recommended way to track and manage service usage.

2.3.6 Service Discovery

For all practical purposes, service discovery is a design time exercise. The primary reasons involve semantics and integration.

Semantics specify exactly what a service does, how it behaves in normal and exception cases, and the meaning of each data element sent and received. This is something that requires thought and reasoning. The SOA infrastructure helps in this regard by providing a place to manage all the information required to make semantic-related decisions.

Integration is simply the act of coding or configuring the interactions involved with service invocation. It includes the steps necessary to prepare for service invocation, as well as the steps to process normal and exception case results. While tooling can help accomplish some of this through configuration, it is difficult to imagine the entire integration scenario taking place dynamically at runtime. Note that runtime integration is different than runtime binding. Conceptually, runtime binding is the act of connecting a consumer with a provider. The consumer and provider must already be compatible, both in terms of semantics and integration. The connection can be made either through routing decisions or templates (using a service registry). But in either case, the service discovery aspect must be performed first in order to overcome semantic and integration hurdles.

2.3.7 Policy Management

SOA enables policy driven architecture that decouples specific business rules, regulatory compliance checks and security policies to be modeled and deployed independent of the services. This abstraction gives the flexibility to model once and deploy multiple times in addition to allowing centralized control over these business policies. Policies allow the dynamics of the system to be quickly changed and provide visibility into the business by plugging into the SOA management frameworks. SOA infrastructure should include capabilities to manage, distribute and monitor the policies.

2.3.8 Dependency Analysis

As the maturity of SOA increases, the number of services and other interdependent assets will grow rapidly as well. These interdependencies are hard to track if not managed properly. An important part of SOA governance is the ability to assess the impact in development and operational environments as a result of a change to a given asset. Dependency analysis is beneficial as the number of services increases, the relationships become more complex, and the need to revise or retire services arises. It makes it easier to perform impact analysis when changes need to occur. This goes both ways - one may need to understand the dependencies a service has, if the service needs to be changed, moved, or virtualized; or one may need to understand what

services depend on a particular resource or service, if the resource or service needs to be modified, retired, or moved.

2.4 Management

Service management refers to the configuration of SOA Infrastructure to control the runtime aspects of the deployment. The following capabilities are commonly looked for when evaluating service management infrastructure:

- Service Level Agreements (SLA) Management
- Logging and Monitoring
- Versioning Support
- Resource Browsing
- Environment Propagation

These capabilities are discussed in detail below.

2.4.1 Service Level Agreements (SLA) Management

The ability to configure service level agreements (SLAs) on service end points and infrastructure-provided services on the following attributes is always beneficial:

- Average processing time of a service
- Processing volume
- Number of errors, security violations, and schema validation errors

The ability to configure alerts for SLA rule violations as well as enable or disable Services based on this data is also very useful.

2.4.2 Logging

The ability to configure the level of logging and auditing is an important capability of the service management infrastructure.

2.4.3 Versioning Support

The infrastructure should provide the ability to stage the deployment of multiple versions of a service which will allow multiple versions of a service to be available at runtime. The ability to provide routing and transformation between versions is an essential infrastructure feature that ensures that the right version of the service is consumed. Different consumers might want to access different versions of the service for various reasons and the infrastructure should route the requests appropriately.

2.4.4 Resource Browsing

SOA infrastructure is made up of several moving parts, such as service endpoint information, schemas, transformations, WSDLs, and policies. A certain level of automation is required to ensure that the resources are detected and auto-configured. All resources should be kept in sync and should act in unison to run the service successfully. That makes it important to be able to browse the resources and their configurations exposed through the SOA Infrastructure so that they can easily be registered and managed.

2.4.5 Environment Propagation

Since SOA Infrastructure is heavily configuration based, the ability to propagate the configuration information from environment to environment is highly important. This should include the ability to override system specific settings that vary from environment to environment (such as host names, etc.).

2.5 Monitoring

The monitoring related SOA capabilities include:

- Runtime Service Usage Tracking
- Exception Management
- Performance Management
- SOA Dashboard

These capabilities are discussed below.

2.5.1 Runtime Service Usage Tracking

In complicated SOAs with potentially hundreds (or even thousands) of participants it becomes increasingly important to have a centralized point of tracking and analyzing data related to the operation of both SOA Infrastructure and the services participating in the SOA. The SOA monitoring infrastructure analyzes, stores, and acts upon runtime data to ensure the optimal operation of the runtime environment. It also provides the information back to the operations team which enables informed decisions to be made about scaling the infrastructure or whether there is room to expand the usage of the infrastructure and its services across additional applications and service consumers. Service usage tracking gives the ability to track what services are being accessed and by which consumers and version usage tracking gives the ability to track which versions of a service are being used and by whom.

2.5.2 Exception Management

The SOA infrastructure should be capable of monitoring and analyzing the exceptions that occur at various parts of the infrastructure. There are several types of exceptions that need to be handled by the infrastructure, including:

- Functional exceptions
- Business exceptions
- Service availability exceptions (show how often the service is available or unavailable)
- Security violations (show the attempts to use a service without proper access rights)

2.5.3 Performance Management

Response-time data for each service and the system load, optionally plotted over time of day and days of the week, would be vital information for managing the performance of the system. The infrastructure should aggregate and provide the performance metrics to assist troubleshooting and process improvement activities. The system performance should be closely monitored and proactively acted upon to ensure that the Services are highly available and meet the stringent SLA requirements.

2.5.4 SOA Dashboard

A SOA Dashboard is a system that collects data from several service based systems, aggregates them and measures the results against metrics in order to provide an interpretation in the form of control panels and reports. By defining a dashboard based on metrics around SOA, the system will provide the management with the means to monitor the SOA adoption process and identify strengths and weaknesses in the process and organizational areas to act upon accordingly.

2.6 SOA Security

Due to its inherent distributed nature, SOA can greatly complicate the security landscape of an IT organization. It stands in opposition to silo'ed applications, which are typically secured by adding layers of protection around the perimeter, in favor of distributed functions and data, which are much more open, and potentially vulnerable. The SOA Security infrastructure is meant to address this problem. It extends from the security infrastructure already in place to meet the challenges presented by the adoption of SOA.

The Security infrastructure must address the ability to secure messages, e.g., message level security, as well as the ability to ensure that functions and data are accessible to the correct audience and under the right conditions. It must do so in a way that is scalable and yet manageable. This involves the unification of assets that drive security decisions, such as LDAP directories, databases, etc., as well as the centralization of policy management.

The Security infrastructure provides the standard capabilities, such as authentication, authorization, encryption, credential mapping, non-repudiation, confidentiality etc., to the other SOA components in addition to the SOA specific capabilities. These capabilities may be offered as services, much like other types of Services that can be discovered, versioned, and invoked through standards-based interfaces. Or, they may be accessed by application containers through low level APIs.

SOA security is covered in more detail in the *ORA Security* document. The SOA specific capabilities of the Security infrastructure are summarized below.

2.6.1 Standards based security

The SOA security infrastructure must enable choice and interoperability through the support of industry accepted security standards. A number of security standards such as WS-Security, WS-SecurityPolicy, XML signatures, XML encryption, and SAML, enable security interoperability between the disparate components of the SOA infrastructure. Standards also ensure that the security technologies are compatible with existing security products and capable of being leveraged across a diverse array of web servers, application servers, and custom applications built in various languages.

2.6.2 Security policy provisioning

The security infrastructure must efficiently distribute incremental updates to policy and configuration data and ensure synchronization across the enterprise. The infrastructure must allow provisioning of security policies that control access and authorization.

2.6.3 Distributed policy decision-making and enforcement

The SOA security infrastructure must provide a means for policy decisions to be defined centrally and enforced locally to meet performance requirements. Some of the security decisions can be pushed to the periphery through an agent-based architecture to improve the performance of the transaction and integrity the system. Distributed policy decision points and policy enforcement points allow a dynamic and flexible architecture that can respond to the needs of the business much faster.

2.6.4 Security Management

SOA infrastructure should provide the following Security Management capabilities:

- The ability to manage policies for authentication, encryption and decryption, and digital certificates as defined in the Web Services Security (WS-Security) specification.
- The ability to enable traditional transport-level security for HTTP and JMS protocols by utilizing SSL.
- Support for one-way and two-way certificate based authentication, as well as providing support for HTTP basic authentication is needed to support basic security requirements.

2.6.5 Centralized security management

The security infrastructure should provide an integrated enterprise policy "system of record" that eliminates fragmentation across disparate applications. The security management component of the SOA Infrastructure allows configuration of the policies around SOA resources centrally at design-time, and then publishes the policies to the relevant security infrastructure responsible for enforcement at runtime.

2.7 Other Complementary Capabilities

SOA is an architectural style that can be applied to various technology strategies. A comprehensive IT infrastructure will require much more than the SOA capabilities. A number of technology strategies like BPM, EDA, BI and Enterprise 2.0 can take advantage of SOA and provide complementing capabilities that help complete the enterprise picture. These capabilities are covered in separate technology-related perspectives but it is important to understand that there are complementary and overlapping capabilities between SOA and the other technologies.

For example, SOA asset management and BPM asset management are quite similar and the same asset management solution can be applied to both. Business processes can be implemented by orchestrating services and they can be exposes as services for external consumption. BPM can leverage the service discovery and routing capabilities of SOA to achieve that.

Similarly SOA can take advantage of the messaging and eventing capabilities of the EDA infrastructure and EDA can invoke the Services to perform an unit of work on the occurrence of an event.

2.8 SOA infrastructure and SOA Maturity

As the adoption of SOA increases, infrastructure will play a key role in advancing the level of maturity. It supports the activities and provides capabilities that are necessary to "raise the bar" in terms of what can be accomplished through SOA. Though these capabilities can be custom developed, it is generally more cost effective to buy the

infrastructure rather than build it. Standards make it possible to mix and match best of the breed products across vendors and avoid vendor lock-in. They also make it possible, and even highly likely, for infrastructure to be acquired and deployed as the need arises as opposed to all at once.

Figure 2–3 SOA Infrastructure and SOA Maturity

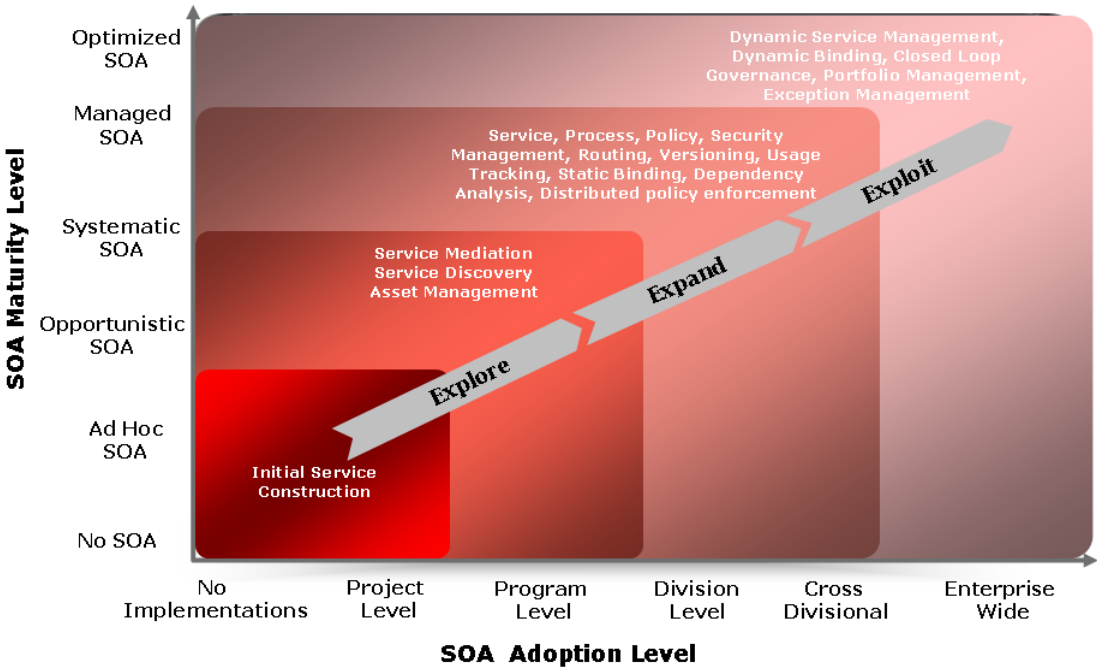


Figure 2–3 shows the typical set of capabilities required as the SOA maturity levels increase and SOA adoption widespread in the organization. The required capabilities would also depend on the specific needs of the projects and services being implemented. As shown above, maturity depends on the ability to define, manage, and optimize the SOA environment. It is also beneficial to ensure a standardized approach is taken as the breadth of SOA increases.

Initially, at the project-level, the benefits of SOA infrastructure are minimal. As SOA becomes adopted at the program level, the need for asset management, discovery, and mediation become important. The definition of service engineering processes, categorization of assets into a taxonomy, and establishment of mediation patterns must emerge and become instituted; otherwise the proliferation of ad-hoc approaches will create chaos.

As SOA moves out across divisional boundaries, policies for security and management gain in importance. Centralized monitoring and management become more and more necessary in order to support expansion in a reliable and secure manner. The growth of the service portfolio will promote building new business capabilities by assembling new services. Service routing, versioning, usage tracking, dependency tracking and distributed policy enforcement capabilities would be most used at this level of maturity.

Further benefits can be realized as the environment is optimized for efficiency and performance. As organizations reach "SOA nirvana", they would be using the capabilities that assist with continuous improvement of the services and business value. Dynamic service binding, dynamic service management and closed loop governance are some of the capabilities that will be in high demand at that level of

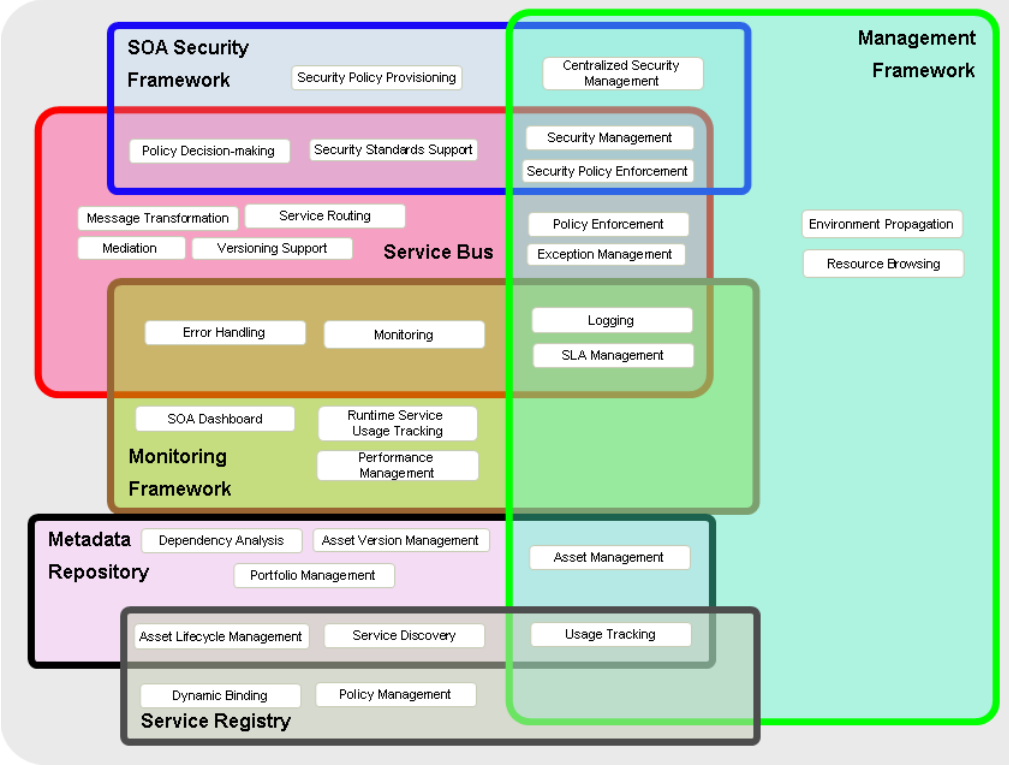
maturity. Infrastructure provides the means to gather key indicators to make optimization possible, either through manual or automated means.

3

SOA Infrastructure Logical View

This section explores the logical architecture of SOA Infrastructure. The various layers and components of the architecture are placed logically to illustrate the relationship between these components.

Figure 3–1 SOA Infrastructure Capabilities and Logical Components



The SOA infrastructure capabilities described in the previous section can be grouped into the following logical components as shown in Figure 3–1.

- Service Bus
- SOA Security Framework
- Service Registry
- Metadata Repository
- Monitoring Framework

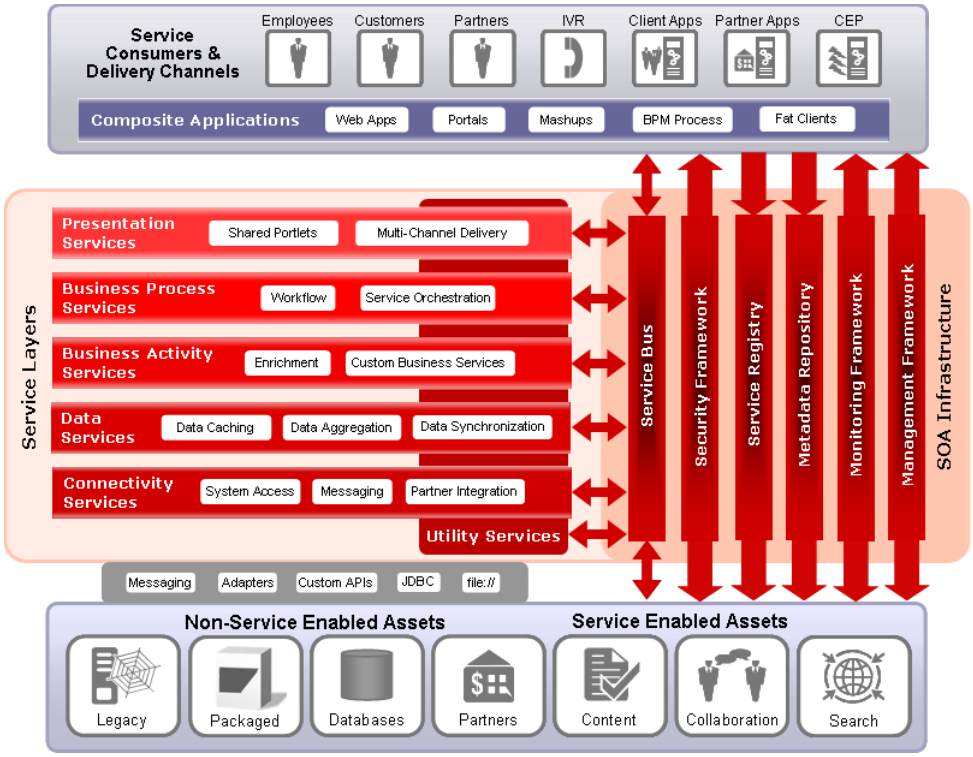
- Management Framework

As shown in the diagram, the logical components overlap with similar capabilities. Deciding which component to use for a given capability is an architecture trade-off. The reference architecture and SOA guiding principles play a key role in determining which logical component should be used and how. Most SOA infrastructure components would have some form of management capabilities. Some are closely related like the metadata repository and service registry. Monitoring features are also generally built into each logical component.

These logical components are meant to satisfy the conceptual capabilities required by a SOA. They may be individual products or a combination of products and technologies that satisfy a logical need. For example, security, monitoring, and management frameworks may be comprised in a number of ways, depending on the needs of the enterprise, using various products and/or combinations of products. A recommended approach using Oracle products is provided in Section 4.1.

The logical Reference Architecture that shows the Services layer, consumers, providers and the SOA infrastructure components is depicted in Figure 3–2:

Figure 3–2 SOA Infrastructure Logical View



The Service Bus is shown including arrows depicting the connections between service layers, service consumers, and service-enabled IT assets. This is a runtime invocation pattern, where the Service Bus provides mediation between service producers and consumers. Other components of the SOA Infrastructure are shown without connecting lines and arrows. This is due to the complex nature of interplay between these components and everything else in the diagram. A single view illustrating all possible interactions would be unreadable, therefore separate diagrams will be used to illustrate these relationships.

It is also important to note that infrastructure provides value in multiple phases of the software lifecycle. This document will describe the role of infrastructure in design time, runtime, and management capacities.

Further, the above diagram depicts a single segment of an SOA deployment. Segmentation across message bus deployments will be discussed in more detail in Chapter 5.

The logical infrastructure components shown in the previous diagram comprise the current realm of infrastructure for SOA. These components are described in greater detail in the following sections. An Oracle product, or combination of products, may be used to address each component. The following sections will describe what the component is, and Section 4.1 will describe how products may be mapped to each component.

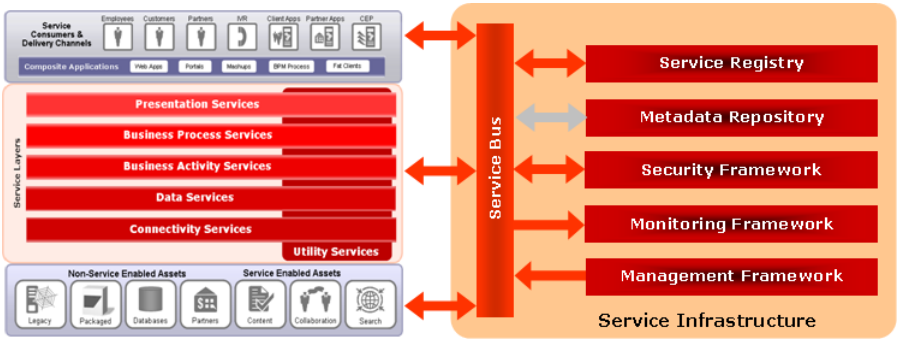
3.1 Service Bus

The service bus acts as the conduit for communication between all participants of the SOA (service consumer and service producers). This intermediary provides the ability to achieve loose coupling and a higher level of flexibility as integration points between consumer and provider can be configured at runtime rather than hard coded. This technique also isolates service consumers from minor changes in the service provider.

The service bus also provides the capabilities to incorporate some of the strategies that may have been introduced by the SOA planning organization of an enterprise such as service versioning strategies, message routing architecture, or the creation of a service network. The service bus provides a flexible and open runtime architecture allowing the enterprise the freedom to enable their existing investment in IT technology as well as be positioned to manage and rollout their maturing SOA effectively.

Figure 3–3 illustrates the interactions between infrastructure components and logical relationships. Bidirectional arrows to non-infrastructure components represent typical runtime service interactions. Bidirectional arrows to the Registry and Repository indicate design-time importing and exporting of service data. This is necessary to keep the infrastructure synchronized.

Figure 3–3 Service Bus Logical Relationships



A key function of the service bus is its ability to mediate between endpoints (consumers and producers). Mediation may be in the form of transport, message, and security exchange. This allows endpoints employing different technologies to interact without the need to construct unique integration code, i.e., point-to-point custom connections. In addition to mediation, the service bus must also support transformation and routing.

The capabilities described are typically provided by the Service Bus. Not every enterprise will require all of the above capabilities for use within their SOA but it is important to understand their needs especially if they have selected a service bus already. In this case they need to be aware of the limitations of the product with respect to their SOA needs.

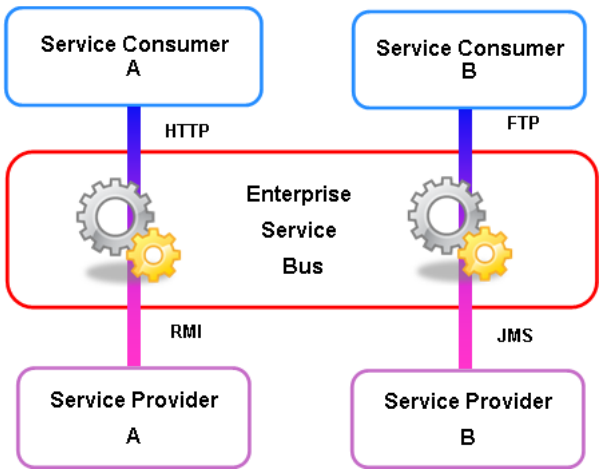
3.1.1 Mediation

Service Bus support several types of mediations including transport mediation, message mediation and security mediation.

3.1.1.1 Transport Mediation

The set of supported transport protocols is important to consider when selecting the SOA Infrastructure for composition. The Service Bus supports the common routing transports that may typically be used to invoke services or pass messages within an SOA. A sample list of these transports has been listed in Section 2.2.1.2.

Figure 3–4 Transport Mediation



As shown in Figure 3–4, the service bus can mediate transports by providing a proxy endpoint of one type and invoking services using another. For example, a service offered via Tuxedo, or MQ Series might be accessed via a JMS client. This allows a Java client to invoke services without the need to install additional client libraries and hard code a specific interface. The service bus would handle all routing, transformation, and security issues involved in the mediation and return results of the service to the consumer.

3.1.1.2 Message Mediation and Transformation

Similar to transport differences, there may be differences in the way messages (request and response data) are formatted between services. Messages of various formats and types must be consumable by the Service Bus. The following are common message formats that may typically be utilized within an SOA:

- SOAP and SOAP with attachments (SOAP that is or is not described by WSDL)
- XML and XML with attachments (XML that is or is not described by a WSDL or a schema)
- Text

- JMS (with headers)
- MFL (Message Format Language)
- Raw Data (opaque data which is non-XML data for which there is no known schema)

The following message delivery models should also be available:

- Synchronous
- Asynchronous
- Synchronous-to-Asynchronous bridging
- Publish/Subscribe
- One-way (Fire and Forget)
- Reliable messaging

The rules that define acceptable message and transport structures should be contained within the SOA Reference Architecture. These rules should describe which combinations to use under which circumstances. It is then up to the SOA Infrastructure to support them, and more specifically the Service Bus. The Service Bus must provide the ability to convert from one to another depending on usage scenarios. Converting messages may be as simple as inserting text taken from one element into another. Or, it may involve a complex series of transformation steps.

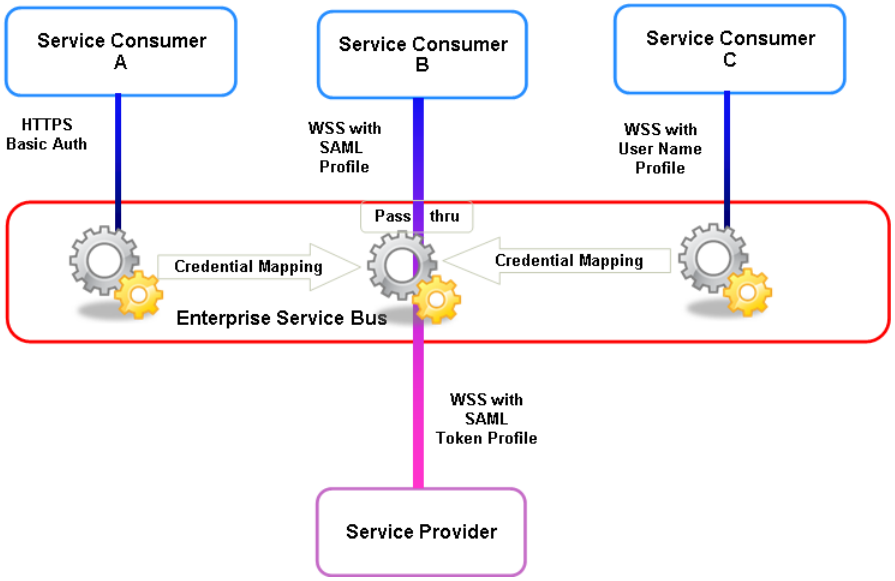
The ability to transform messages based on configuration is a powerful capability, and provides a great deal of flexibility when composing and designing services and applications. The following transformation capabilities should be examined when selecting the SOA Infrastructure:

- Transform messages based on the target service
- Transform messages based on XQuery or XSLT
- Support transformations on both XML and MFL messages
- Message enrichment capabilities
- Callouts
  - Support call out to Web services to gather additional data for transformation.
  - Support call out to Java classes.
- Message validation against schemas

3.1.1.3 Security Mediation

Service bus could provide the security mediation capability. One example of security mediation, illustrated in Figure 3–5, involves the conversion (mapping) of credentials from one form to another.

Figure 3–5 Security Mediation



Each consumer sends a specific type of credential. The actual service implementation requires a specific credential; therefore any other credential received must be mapped to what the backend wants. In this example, all endpoints happen to be using HTTP(S) as a transport. The diagram could be extended to show different transports, each with their own proprietary security technology. In this way transport, message, and security mediation could all be required in order to connect consumers and producers.

3.1.2 Routing

Routing is a fundamental capability of the service bus that enables loose coupling. It can be used to direct requests based on business rules, SLAs, maintenance windows, service versions, etc.

The ability to route messages according to content or header based routing policies or callouts to external services is also beneficial. Routing policies should apply to both point-to-point and one-to-many (publish) messaging models. Routing based on expected and unexpected error conditions is also very useful.

3.1.3 Monitoring, Management and Security

Given the unique position of the Service Bus as an intermediary, it may be used for a number of supplementary functions. Monitoring, management, and security are particular areas of interest that are described in Section 3.4, Section 3.5 and Section 3.6. This is an example of where infrastructure concerns and products tend to overlap. The primary purpose of the Service Bus may be to mediate service interactions, however it may be used to monitor, manage, and secure them as well.

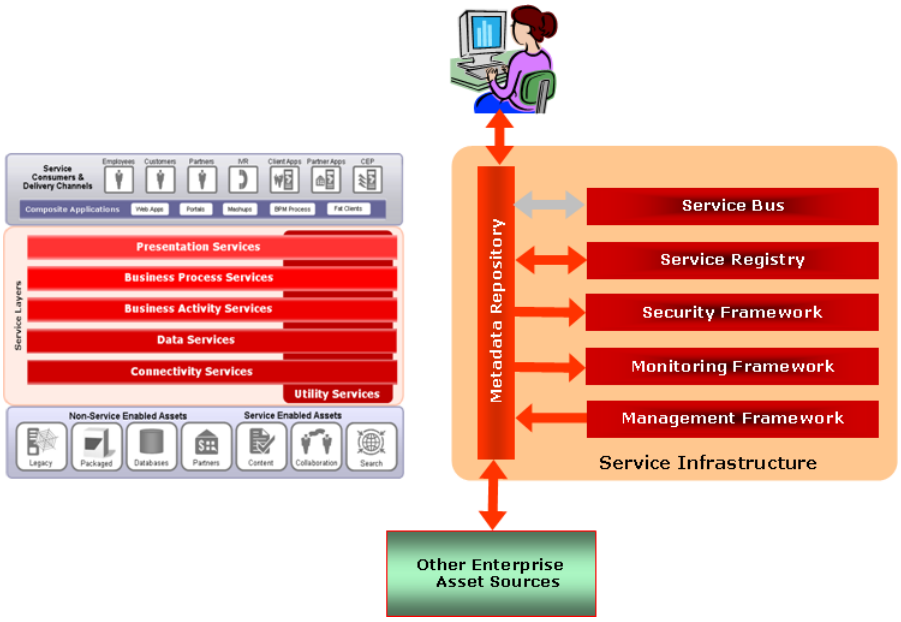
The interaction between the Service Bus and Security Framework is bidirectional as shown in Figure 3–3. The Security Framework provides functions, such as authentication, authorization, etc., that the Service Bus may leverage. It may also push entitlements policies down to the Service Bus so that access control decisions can be made locally. In addition, routing decisions can be made based on entitlements, which are provided by the Security Framework.

Management and monitoring are generally one way interactions. The Service Bus sends monitoring data to the Monitoring Framework. It receives management instructions from the Management Framework.

3.2 Metadata Repository

The primary focus of the Metadata Repository (aka Enterprise Metadata Repository or Enterprise Repository) is design-time, and it usually has no role in the runtime environment of most SOA deployments. The metadata repository interactions are shown in Figure 3–6. The metadata repository is primarily a human interface for asset capture and presentment. It has integration with the service registry to promote the service interfaces and with the security framework for repository security like authentication and access control. It also has integration with other enterprise asset sources like **Source Code Management** (SCM) tools and file servers.

Figure 3–6 Metadata Repository Logical Relationships



Core capabilities of the metadata repository include:

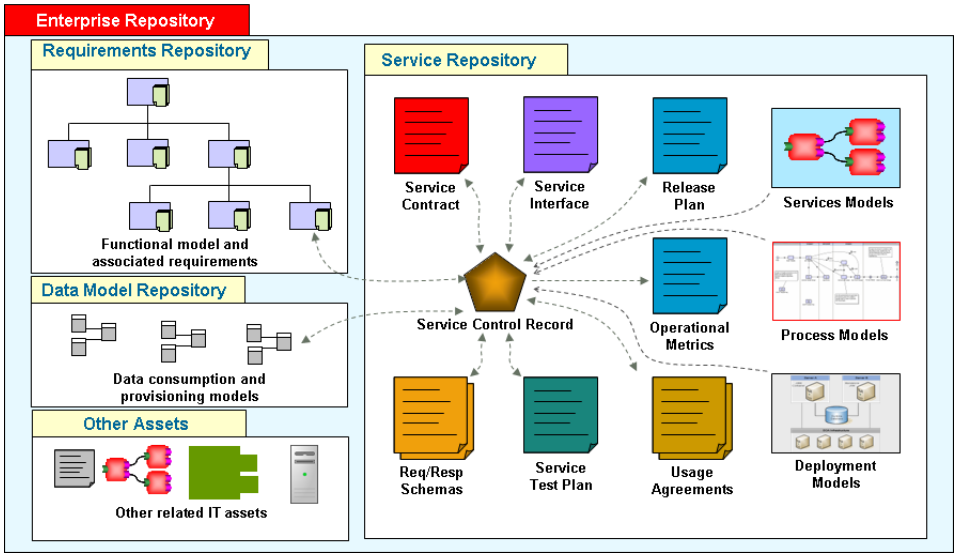
- Asset Management
- Asset Lifecycle Management
- Usage Tracking
- Service Discovery
- Version Management
- Service Taxonomy
- Dependency Analysis
- Portfolio Management

In addition, the metadata repository also offers service publication, compliance check, policy check and reporting capabilities. These capabilities enable it to completely fulfill

the needs of a metadata repository. It also plays a part in SOA monitoring and management.

The metadata repository provides much more than storage for web service interface definitions (WSDL). The repository provides a centralized holding area for a great deal of SOA related information that will be utilized at design time to construct additional services and applications. The repository also provides the primary means for service discovery. In many ways, the service repository can be utilized as the center point for service oriented design.

Figure 3–7 Metadata Repository



As shown in Figure 3–7, the repository can be used to store and link together service artifacts, models and other assets. These artifacts may be related to services design, such as service and process models; service development, such as contracts, interfaces, and schemas; and service maintenance, such as **usage agreement**s and operational metrics. The repository provides a means to organize services into a taxonomy, link related artifacts together, and identify dependencies between services.

3.3 Service Registry

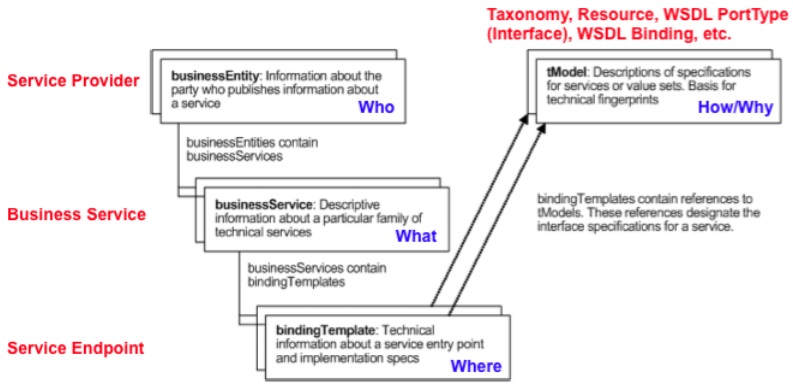
The Service Registry provides a standards-based runtime interface for service assets. Service Registries are generally UDDI compliant registries that provide runtime location transparency and dynamic binding abilities. The registry also federates runtime metrics for closed loop governance.

The Service Registry has many potential purposes, however, most are better addressed today through other means. For example, the registry can be used to achieve loose coupling by offering dynamic binding to service endpoints. Dynamic binding refers to the act of connecting a service consumer to a producer at runtime. The consumer accesses the registry when invoking a service and the registry determines which service producer to use. The registry returns endpoint information to the consumer, who then accesses the appropriate endpoint.

Dynamic binding satisfies the need for loose coupling; however it requires two round trip interactions - one between the consumer and registry, and the other between the consumer and producer. It also requires the consumer to request a service using a

template, which the registry uses to locate a suitable service provider. This programming model is much more complex than coding directly to a service endpoint. A better model for loose coupling is the use of a Service Bus. This component logically sits between the consumer and producer, acting as a man-in-the-middle. The consumer invokes proxy services hosted by the service bus, which in turn access the service provider on behalf of the consumer. This architecture pattern provides many benefits, which has been described in Section 3.1.

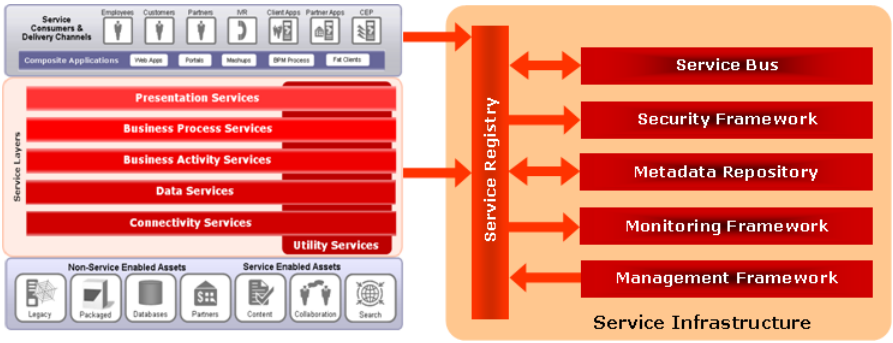
Figure 3–8 Service Registry Entities



The registry can also be used to locate services at design-time. It typically supports the UDDI protocol for service lookups, and offers interactive search capabilities. The registry organizes assets into a taxonomy of entities and tModels, as shown above. This allows it to organize services into a taxonomy and make them accessible to runtime queries. One could consider the taxonomy and search to be service discovery capabilities. However, since the structure of the registry is generally no match for what can be supported by the repository, it has been eclipsed by the repository for design-time discovery and metadata storage.

The runtime access feature does provide a benefit, even if it is not used for dynamic binding and loose coupling. It can be used to store runtime policies, such as security policies. Policies can be associated with one or more services, and indirectly accessed via the registry. This makes the registry part of the security infrastructure.

Figure 3–9 Service Registry Logical Relationships



At runtime, the registry may be accessed by service consumers, either to obtain service bindings or policy references for services they are attempting to invoke. This is

represented by arrows from the Service Consumer, Services, and Service Bus boxes to the registry. The registry may leverage the Security Framework to secure its functions, and it may send events to the Monitoring Framework. The Management Framework may be used to control which services are listed in the registry and which endpoints to use for runtime dynamic binding.

At design-time, the registry may exchange service information with the Service Bus and/or Repository, in order to keep the SOA Infrastructure components synchronized. The exchange may occur in either direction, and it may be manual or automated. The Service Registry can also publish runtime statistics into the repository. This gives Architects the comfort to reuse operational services as they prove that they deliver what the contract promises.

3.4 SOA Security

Due to its inherent distributed nature, SOA can greatly complicate the security landscape of an IT organization. In contrast to silo'ed applications, which are typically secured by adding layers of protection around the perimeter, SOA stands in favor of distributed functions and data, which are much more open, and potentially vulnerable. The Security Framework is meant to address this problem. It extends from the security infrastructure already in place to meet the challenges presented by the adoption of SOA.

The Security Framework must address the ability to secure messages, e.g., provide message level security, as well as the ability to ensure that functions and data are accessible to the correct audience and under the right conditions. It must do so in a way that is scalable and yet manageable. This involves the unification of assets that drive security decisions, such as LDAP directories, databases, etc., as well as the centralization of policy management.

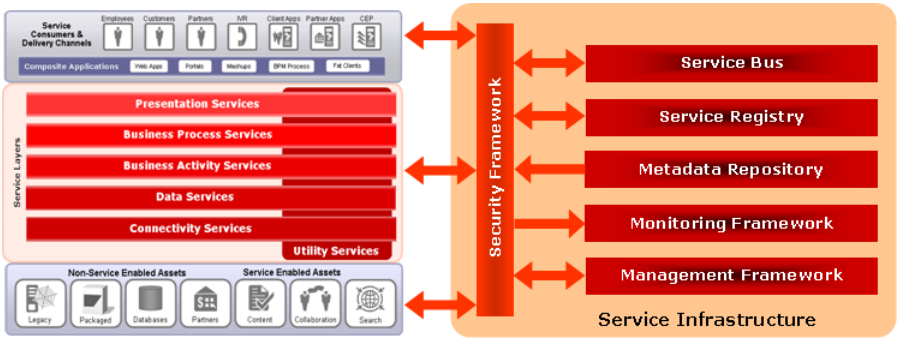
The following ideals apply to the Security Framework in order to fit the needs of SOA:

- **Standards support:** Must enable choice and interoperability through the support of industry accepted security standards.
- **Security policy provisioning:** Must efficiently distribute incremental updates to policy and configuration data and ensure synchronization across the enterprise.
- **Distributed policy decision-making and enforcement:** Must provide a means for policy decisions to be enforced locally to meet performance requirements.
- **Centralized control of security policy and configuration data:** Should provide an integrated enterprise policy "system of record" that eliminates fragmentation across disparate applications. The security management component of the SOA Infrastructure allows you to configure the policies around SOA resources centrally at design-time, and then publish the policies to the relevant security infrastructure responsible for enforcement at runtime.
- **Other Features**
    - **End-to-end coverage:** Security should extend from the perimeter back to everything that plays a role in the SOA.
    - **In transit and at rest:** Messages should be protected while they are persisted in databases, queues, etc., as well as when they are being transmitted on the wire.
    - **Service-oriented approach:** The framework should enable security as a service, provided by consistent and re-usable infrastructure components.

- **Extensibility:** The framework should provide well-defined, "pluggable" interfaces that enable it to be extended to meet future needs without rework.
- **Support for heterogeneity:**The framework must be compatible with existing security products and be capable of being leveraged across a diverse array of Web servers, application servers, and custom applications built in various languages.

Logically, the security framework may interact with all service layers, infrastructure components, service consumers, and IT assets that operate in the SOA environment, as shown below.

Figure 3–10 Security Framework Logical Relationships



The Security Framework provides capabilities, such as authentication, authorization, encryption, credential mapping, etc., to the other SOA components. These capabilities may be offered as services, much like other types of Services that can be discovered, versioned, and invoked through standards-based interfaces. Or, they may be accessed by application containers through low level APIs.

Arrows pointing toward the Security Framework represent interactions where these security capabilities are being used by other components. In addition, the Management Framework provides the ability to manage security policies and policy assignments. Arrows pointing away from the Security Framework indicate places where policy may be pushed out to the other components in support of local access control decision making. In the case of the Monitoring Framework, the arrow represents the flow of security audit events that are actively monitored.

More details of SOA Security is covered in depth in the *ORA Security* document.

3.5 Service Monitoring Framework

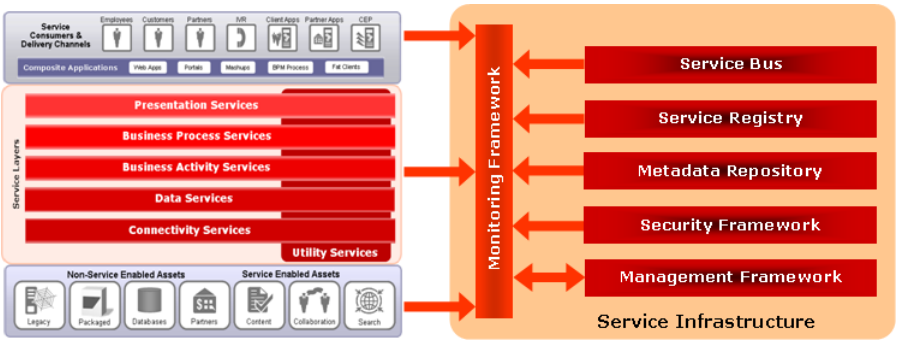
In mature SOAs with potentially hundreds (or even thousands) of participants it becomes increasingly important to have a centralized point of tracking and analyzing data related to the operation of both the SOA Infrastructure and the services participating in the SOA. The Service Monitoring Framework analyzes, stores, and acts upon runtime data to ensure the optimal operation of the runtime environment. It also provides the information back to the operations team which enables informed decisions to be made about scaling the infrastructure or whether there is room to expand the usage of the infrastructure and its services across additional applications and service consumers. Examples of where monitoring can be applied include:

- **Service usage**: The ability to track what services are being accessed and by which consumers

- **Version usage**: The ability to track which versions of a service are being used and by whom
- **Performance**: Response time data, optionally plotted over time of day and days of the week, for each service
- **Exceptions**: Functional (service invocation errors) and business exceptions
- **Availability**: How often the service is available or unavailable
- **Security violations**: Attempts to use a service without proper access rights

As shown below, the Monitoring Framework can receive run time data and events from any component in the SOA environment. SOA Dashboards usually synthesize and present the information in a human friendly format.

Figure 3–11 Monitoring Framework Logical Relationships



The Service Monitoring Framework is usually more than a single product. It can be a combination of infrastructure components to instrument service interactions, as well as functions built into the services platforms that provide audit capabilities, usage metrics, and performance data.

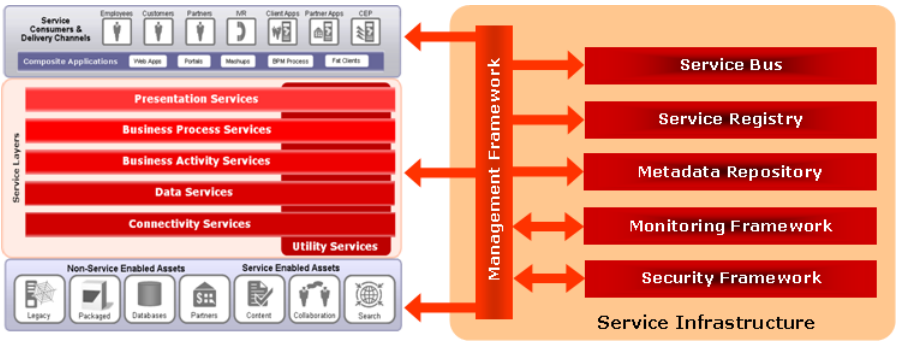
Ideally, the Service Monitoring Framework receives its runtime configuration from the Service Management Framework, which enables it to adapt to changes without the need for coding and redeployment.

3.6 Service Management Framework

Service management refers to the configuration of the SOA Infrastructure to control the runtime aspects of the deployment. The Service Management Framework is also used to activate the configuration at runtime. The following capabilities are commonly looked for when evaluating service management infrastructure:

- Security Support
- Service Level Agreements Management
- Logging and Monitoring
- Versioning Support
- Resource Browsing
- Environment Propagation

Figure 3–12 Management Framework Logical Relationships



The Management Framework is used to control, or manage, various aspects of the SOA environment. Therefore the relationship is generally unidirectional. One exception is the relationship between the Management and Monitoring Frameworks. Monitoring data may feed into the Management Framework in order to automate management activities. For example, runtime performance data may automatically control resource provisioning. Likewise, security events, such as those indicative of hacking, may automatically trigger defensive measures such as the tightening of security policies or the denial of services from a particular consumer.

Another exception to the unidirectional relationship is the interaction between the Management and Security Frameworks. The Security Framework may control access to management functions, and therefore may push access control decisions out to the Management Framework.

4

SOA Infrastructure Product Mapping View

This section describes the physical view of the SOA infrastructure. It begins by mapping the Oracle products to the logical infrastructure components and describes each of the SOA infrastructure products in the context of the SOA capabilities. The Reference Architecture discussed in Chapter 2 shows two distinct parts, Service Layers and SOA infrastructure.

Service Layers are the architectural categorization of Services and relate to the Service implementation platform. In contrast, the SOA Infrastructure relates to the common capabilities required to build an enterprise class SOA platform.

Although the primary focus of this document is the SOA infrastructure, this chapter also provides an overview of the technologies needed to build the Service platform in Section 4.2. Detailed product documentation is available online and the URLs are provided in Appendix A, "Further Reading".

> **Note:** The best and latest source of product information is the online product documentation. Links to the online documentation are provided in Appendix A, "Further Reading". This section presents version specific information like product features and standards support that are current as of the writing of this document. Please refer to the product documentation for your specific version of the product for the latest supported features and standards. The standards stated in this document may be a partial list and some may be partially supported by the products.

4.1 Oracle Product Mapping - SOA Infrastructure

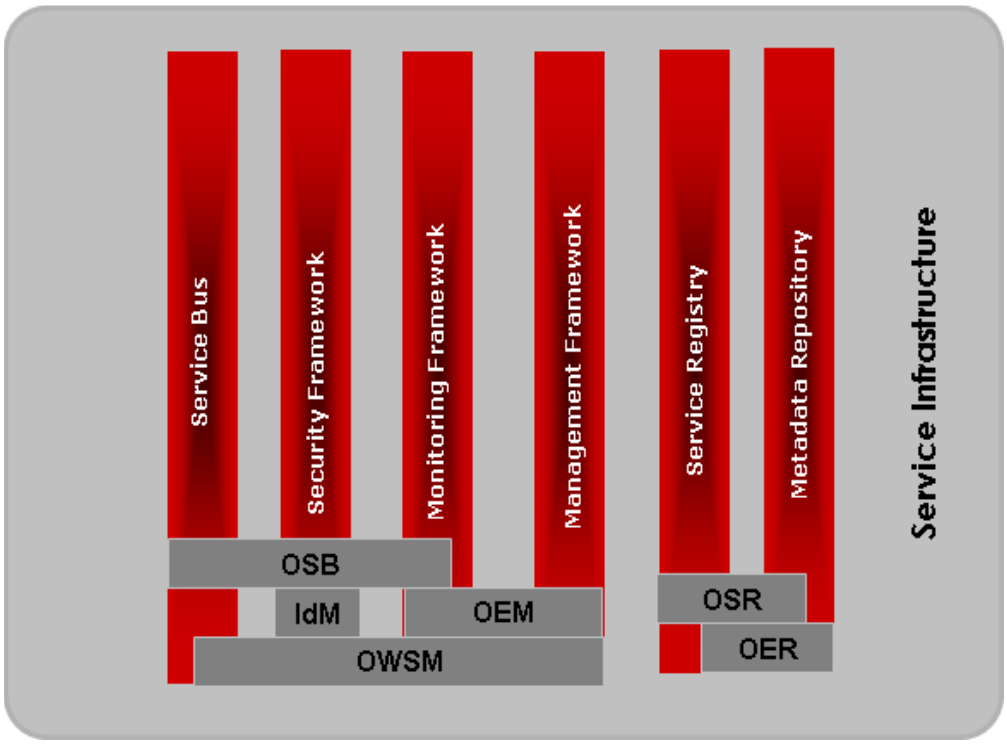
Oracle offers a comprehensive SOA solution through a suite of SOA products. Oracle Fusion Middleware products cover the needs of the SOA infrastructure end-to-end. Figure 4–1 shows the mapping of Oracle products to the SOA logical components.

The products referred in the figure are:

- OSB - Oracle Service Bus
- OSR - Oracle Service Registry
- OER - Oracle Enterprise Repository
- OWSM - Oracle Web Service Management
- OEM - Oracle Enterprise Manager (with SOA management pack)
- IdM - Identity and Access Management

As shown in Figure 4–1, there isn't necessarily a one to one mapping between logical architecture components and products. While some products target a specific logical need, most provide additional features, such as monitoring, management, and security.

Figure 4–1 SOA Infrastructure - Oracle Product Mapping



The centerpiece of Oracle's SOA Infrastructure offerings is the Oracle Service Bus (OSB). This product fully satisfies the requirements of a service bus. It also provides service management, monitoring, and security capabilities in addition to the standard service routing, mediation, and transformation features.

The Oracle Enterprise Repository (OER) provides a great deal of design-time capabilities. It provides a robust and flexible repository for storing, managing, and discovering all types of SOA based assets. OER also provides capabilities for quality assurance and SOA governance enforcement. These may be applied to help manage SOA based development work underway in the organization or enterprise.

Oracle Service Registry (OSR) provides a robust UDDI registry for runtime metadata information including service descriptions (WSDL), and policies (WS-Policy). It may be used to support loose coupling through dynamic service binding if a Service Bus is not present. Please note that the Service Bus would be a basic requirement as SOA maturity increases. OSR is also ideal for providing discovery of services when access to a repository is not permitted.

Oracle Identity Management (IdM) Suite provides centralized security management capabilities for an enterprise wide SOA. IdM provides a comprehensive security solution that covers the full spectrum of identity management and access management. IdM is beyond the scope of this document and is covered in the *ORA Security* document.

Management of services is extremely important in SOA environments, where services are integrated, reused, and constantly changed. Oracle Enterprise Manager (OEM) simplifies monitoring and managing SOA environments. It addresses each of the challenges outlined above by helping model, monitor, and manage the SOA environment.

Oracle Web Services Manager (OWSM) is a Web Services security and management solution that provides the visibility and control required to deploy Web Services into production. OWSM allows companies to define policies that govern Web Service operations such as access, authorization, logging, and load balancing, and then wrap these policies around Web services.

The rest of this section goes through the details of each of these SOA infrastructure products.

4.1.1 Oracle Service Bus (OSB)

The category of infrastructure identified as the Service Bus represents many features including:

- **Mediation** - the ability to connect consumers and producers that may use different transports, protocols, message formats, error codes, etc.
- **Message mediation and Transformation** - the ability to transform messages from one format to another, aggregate data elements, and enhance message content based on business rules
- **Routing** - the ability to route requests to different endpoints at runtime based on mechanisms such as message content, business rules, security constraints, etc.
- **Composition** - the ability to execute multiple steps, possibly including multiple back end requests, based on configuration, to fulfill an inbound service request

Oracle Service Bus (OSB) is designed to meet the needs of an enterprise-class service bus. In addition to the Service Bus features, OSB also provides useful monitoring, management, and security capabilities. This makes it applicable to these SOA Infrastructure components as well.

The OSB acts primarily as an intermediary that takes in messages, processes them to determine where to route them, and transforms them as specified. It receives messages through a transport protocol such as HTTP(S), JMS, File, FTP, and so on, and sends messages through the same or a different transport protocol. Message response follows the inverse path. The message processing by OSB is driven by metadata specified as the message flow definition, which is created at design time, for a proxy service in the OSB Configuration.

OSB is policy driven. It enables you to establish loose coupling between service clients and Services while maintaining a centralized point of control and monitoring.

The following features provided by OSB provide value to the SOA environment:

- Service Routing
- Transformations
- Mediation
- Message Flow Modeling
- Error Handling
- Quality of Service

OSB supports the following industry standards:

Topic	Standards Supported
Web Services	SOAP 1.2, SOAP 1.1, WSDL 1.1, WS-Addressing 1.0, WS-Policy 1.5, WS-Trust 1.3, WS-I Basic Profile 1.0, JAX-WS 2.1, JAXB 2.1
Security	WS-Security 1.1, WS-SecurityPolicy 1.2, WS-SecureConversation 1.3, SAML Token Profile 1.0, SAML 1.1, SAML 1.0, WS-Security Token Profile 1.0, Username Token Profile 1.0, X509 Token Profile 1.0, WS-I Basic Security Profile 1.0
Transport	HTTP 1.1, HTTP 1.0, JMS 1.1

4.1.2 Oracle Enterprise Repository (OER)

Oracle Enterprise Repository (OER) provides the tools to manage and govern the metadata for any type of software asset, from business processes and services to patterns, frameworks, applications, components, and models. OER maps the relationships and interdependencies that connect those assets to improve impact analysis, promote and optimize their reuse, and measure their impact on the bottom line.

Core capabilities of the OER include:

- **Asset Management**
  - The primary function of OER is to provide a means to store metadata in a way that related objects can be linked together and managed.
- **Asset Lifecycle Management**
  - OER includes the ability to manage service status through the notion of registered and unregistered assets.
- **Usage Tracking**
  - OER can be used to house **Consumer Contract**s aka **Usage Agreement**s, linked to services. This makes it easy to not only see what a service does, but who is using it.
- **Service Discovery**
  - OER facilitates service discovery by providing a means to locate potential services either through taxonomy navigation, or direct search.
- **Version Management**
  - The service lifecycle and dependency tracking capabilities of OER make it part of the overall version management tool kit.
- **Dependency Analysis**
  - Along with the ability to navigate taxonomies, OER provides the capability to navigate asset relationships. One of the major benefits of using an enterprise repository is to facilitate impact analysis based on inter-dependency.
- **Portfolio Management**
  - OER, with its ability to classify services into a navigable taxonomy, supports portfolio management. The service portfolio manager can use this tool to track services and service candidates with respect to business functions, processes, and plans for future services.

OER supports the following standards:

Topic	Standards Supported
Web Services	SOAP 1.2, SOAP 1.1, WSDL 1.1, BPEL 1.1, UDDI v3
XML	XSD 1.0, XSLT 1.0
Transport	HTTP 1.1, HTTP 1.0, HTTPS

4.1.3 Oracle Service Registry (OSR)

The Oracle Service Registry (OSR) provides a runtime registry for SOA related data. It provides the core functions required to search, browse, and publish resources through the registry.

OSR fully implements the OASIS UDDI V3 standard. The registry has been designed specifically for enterprise deployment and includes many advanced features that make it easy to configure, deploy, manage, and secure.

Oracle Service Registry offers the following key capabilities:

- **Runtime Service Discovery**
  - OSR allows the other SOA infrastructure components to discover services and to keep the service information up to date.
- **Dynamic binding**
  - OSR allows dynamic binding of services through the UDDI interface. It also automatically notifies registry users about changes to components that they depend on.

In addition to UDDI v3 standars, OSR also supports SOAP 1.2 and WSDL 1.1 standards.

4.1.4 Oracle Web Services Manager (OWSM)

Oracle Web Services Manager (WSM) is a comprehensive solution for securing and managing Services. It allows IT managers to centrally define policies that govern web services operations such as access control (authentication, authorization), logging, and content validation, and then attach these policies to one or multiple web services, with no modification to existing web services required. In addition, Oracle WSM collects runtime data to monitor access control, message integrity, message confidentiality, quality of service (defined in Service Level Agreements - SLAs) and visualizes that information using graphical charts. Oracle WSM brings enterprises better control and visibility over their SOA deployments.

Key features of Oracle WSM include:

- **Policy Management**
  - Policy Manager allows administrators to configure operational rules and propagate them to the appropriate enforcement components across an application deployment of any scale and complexity.
- **Policy Enforcement**
  - Oracle WSM provides two kinds of policy enforcement components: Gateways and Agents. Gateways are self-contained modules that run as independent processes. In contrast, Agents are interceptors that run in the same process as the Service end point.
- **Monitoring, Runtime service usage tracking**

- OWSM offers SLA monitoring features like service invocation monitoring and SLA adherence monitoring.
- Offers data collection and reporting capabilities.
- OWSM monitors authentication/authorization activities for all services and audits security violations per Web service, per operation and per client.
- OWSM monitors business process flows end-to-end and provides alerts via propagation of events.

- **SOA Dashboard**
  - OWSM provides an operational dashboard to display vital monitoring information that include service and security statistics.
- **Mediation**
  - OWSM offers out-of-the-box, native support for multiple transports and messaging models.
- **Routing**
  - OWSM offers agent based routing capabilities. It can do content-based message routing and attachment-based content routing.

OWSM supports the following standards:

Topic	Standards Supported
Web Services	SOAP 1.2, SOAP 1.1, WSDL 1.1, WS-Policy
Security	WS-Security 1.0, WS-SecurityPolicy 1.2, WS-SecureConversation 1.3, SAML Token Profile 1.0, SAML 1.1, SAML 1.0, WS-Security Token Profile 1.0, Username Token Profile 1.0, X509 Token Profile 1.0, Encryption algorithms: AES-128, AES-256, 3-DES, Message Digests: MD5, SHA-1, XML Signature, XML Encryption, PKI: RSA v1.5, DSA, JKS
Transport	HTTP 1.1, HTTP 1.0

4.1.5 Oracle Enterprise Manager for SOA (OEM)

Oracle Enterprise Manager is a comprehensive IT management solution that makes IT infrastructure management simpler, leading to better cost control. Enterprise Manager enables management of single instances of Oracle SOA suite, Database, Application Server, or Collaboration Suite using standalone consoles. Enterprise Manager Grid Control is Oracle's single, integrated solution for managing all aspects of the Oracle Grid infrastructure and the applications running on it. .

The Management Pack for SOA delivers comprehensive management capabilities for a Service-Oriented Architecture-based (SOA) environment. By combining SOA runtime governance, business-IT alignment, and SOA infrastructure management with Oracle's rich and comprehensive system management solution, Enterprise Manager Grid Control significantly reduces the cost and complexity of managing SOA-based environments.

Oracle Enteprise Manager Grid Control - Composite Application Monitor and Modeler (CAMM) analyzes J2EE and SOA applications to capture the complex relationships among various application building blocks in its AppSchema model. Using the insights stored in AppSchema, CAMM is able to deliver an environment that self-customizes out-of-the-box, evolves with change, minimizes expert

involvement, and delivers a holistic, service-oriented view across heterogeneous environments.

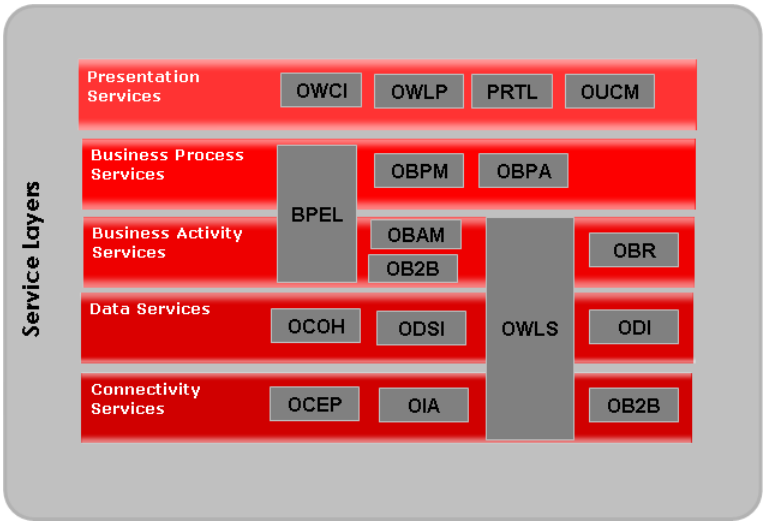
The key features of the Oracle Enterprise Manager for SOA include:

- **SLA Management**
  - Thresholds, alerts, notifications, and SLAs in OEM on external metrics
  - Business KPIs can be used with system metrics to compute SLA
  - OEM Services dashboard displays SLA compliance
- **Logging, Monitoring**
  - OEM provides administrators managing SOA environments with a consolidated browser-based view of the entire enterprise, thereby enabling them to monitor and manage all of their components from a central location.
- **Performance Management**
  - OEM measures and monitors the availability and performance of services for historical trending, troubleshooting, and root cause analysis purposes.
- **Environment Propagation**
  - OEM collects configuration information for the server, domains, and processes. It automates the deployment of resources. Scheduling capability is also provided by OEM.
- **SOA Dashboard**
  - Using EM Grid Control's graphical monitoring dashboards, administrators, managers, and business owners gain real-time understanding of the status of services in a single view. Monitoring dashboards provide top-level views of individual or groups of services, as well as system-level views describing the relationships and status of IT system components.

4.2 Oracle Product Mapping - Service Platform

Services that belong to the layers discussed in Chapter 2 are developed and deployed on various platforms referred to as the Service platform. Oracle offers a slew of technologies suitable for building one or more of the layers of these Services. Figure 4–2 maps the Oracle products to the SOA service layers.

Figure 4–2 Service Platform - Oracle Product Mapping



The products referred in Figure 4–2 are listed by the respective layers below:

- **Presentation Layer** :
  - OWCI - Oracle Web Center Interaction
  - PRTL - Oracle Portal
  - OWLP - Oracle WebLogic Portal
  - OUCM - Oracle Universal Content Mangement
- **Business Process Layer** :
  - BPEL - BPEL Process Manager
  - OBPM - Oracle Business Process Management
  - OBAM - Oracle Business Activity Monitoring
  - OBPA - Oracle Business Process Analysis Suite
- **Business Activity Layer** :
  - BPEL - BPEL Process Manager
  - OB2B - Oracle B2B Integrator
  - OBR - Oracle Rules
  - OWLS - Oracle WebLogic Server
- **Data Services Layer** :
  - ODSI - Oracle Data Services Integrator
  - ODI - Oracle Data Integrator
  - OCOH - Oracle Coherence
  - OWLS - Oracle WebLogic Server
- **Connectivity Services Layer** :
  - OCEP - Oracle Complex Event Processing
  - OIA - Oracle Integration Adapters

- OB2B - Oracle B2B Integration
- OWLS - Oracle WebLogic Server

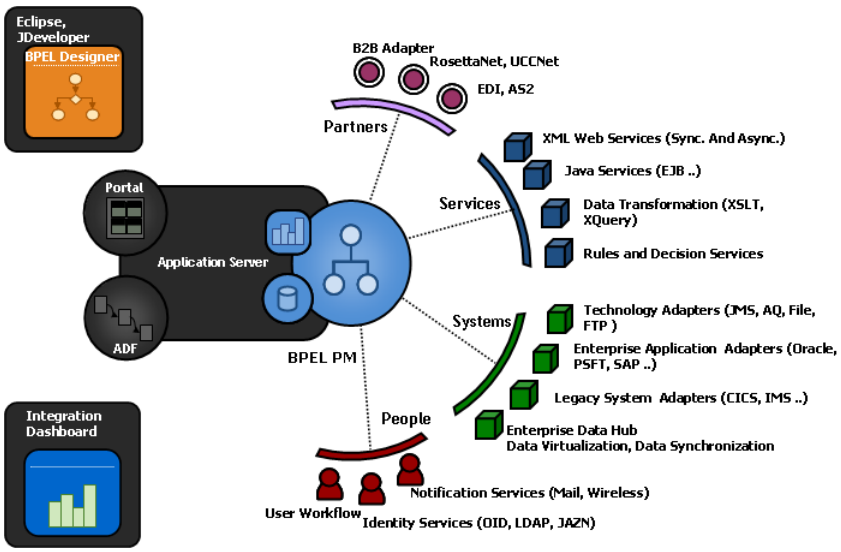
These products are briefly discussed in the following sections. Most of these technologies are covered in detail in the respective Enterprise Technology Strategies (ETS) documentation. A list of these documents can be found in Appendix A, "Further Reading". For example, the products mapped to the business process layer such as BPEL PM, OBPM and OBAM are covered in the BPM ETS Perspective documentation set.

4.2.1 BPEL Process Manager (BPEL PM)

BPEL PM is a business process orchestration engine that implements the **Business Process Execution Language** (BPEL). BPEL-PM provides the ability to quickly build and deploy orchestrations and business processes in a standards-based manner.

Section 5.2.2 compares and contrasts Orchestration and Mediation. BPEL PM plays a key role in providing orchestration capabilities. BPEL PM provides a platform to build business process and business activity services. It can be used to build composite services and business processes by orchestrating existing Services and by implementing new business logic. BPEL PM has built-in integration services that allow connectivity to a number of legacy systems through adapters and native protocols.

Figure 4–3 BPEL Process Manager



As shown in Figure 4–3, BPEL provides a platform and capabilities to build versatile Services with partner, services, system and people integration. BPEL PM's built-in integration services enable developers to easily leverage advanced workflow, connectivity, and transformation capabilities from standard BPEL processes. These capabilities include support for XSLT and XQuery transformation as well as bindings to hundreds of legacy systems through **JCA** adapters and native protocols. Human workflow services such as task management, notification management, and identity management are provided as built-in BPEL services to enable the integration of people and manual tasks into BPEL flows. The extensible WSDL binding framework enables connectivity to protocols and message formats other than SOAP. Bindings are available for JMS, email, JCA, HTTP GET, POST, and many other protocols enabling simple connectivity to hundreds of back-end systems.

BPEL PM supports several open standards in addition to BPEL4WS. These standards include WSIF, WSDL, SOAP, UDDI, JMS, WS-ReliableMessaging, JCA, XQuery, XPath, XSLT and WSIL.

4.2.2 Oracle Business Activity Monitoring (OBAM)

OBAM is a complete solution for building interactive, real-time dashboards and proactive alerts for monitoring business processes and services. It enables business operations to analyze events as they occur through complex event processing and act on current conditions.

OBAM captures **Key Performance Indicator**s (KPI) and Service Level Agreement (SLA) information from the business processes and Services and present it in a format suitable for analysis. OBAM provides agility and transparency by allowing the business users to dynamically change rules without redeploying process.

Business Process Services sometimes generate business exceptions that need to be handled appropriately. OBAM provides a loosely coupled architecture to proactively detect and manage such exceptions.

Business Process Services can also be built dynamically from process fragments using OBAM's dynamic process assembly feature. This allows users to make decisions based on realtime data.

4.2.3 Oracle Business Process Analysis (OBPA)

OBPA provides tools for business process modeling, design, simulation and optimization. OBPA is a design time infrastructure components that allows business processes to be designed and modelled using **Business Process Modeling Notation (BPMN**) and deployed in BPEL PM as BPEL processes. OBPA along with BPEL PM and OBAM provide closed loop BPM capability in the BPM lifecycle.

4.2.4 Oracle Business Process Management (OBPM)

OBPM is a complete set of tools enabling collaboration between business and IT to create, automate, execute, and optimize business processes. Human centric and document centric business process services can be built efficiently using OBPM. The processes could be developed as persistent and long-running or transactional. OBPM also provides design time tools to analyze, design, simulate and optimize the business processes.

In OBPM, business processes are built using the XML Process Definition Language (**XPDL**) and **BPMN** standards.

4.2.5 Oracle Business to Business Integration (OB2B)

OB2B provides a single integrated solution for rapidly establishing online collaborations and automated processes with your business partners. It provides capabilities to manage trading partners and documents with support for a variety of information exchange mechanisms.

In addition to the gateways that provide connectivity to trading partners, OB2B also provides other capabilities to build business activity services. It can easily be integrated with other Oracle SOA products like OSB and BPEL PM to provide an end-to-end business process implementation platform. OB2B provides connectivity services by supporting a number of adapters including EDI and RosettaNet adapters.

OB2B supports a number of industry standards including: EDI, UCCnet, RosettaNet, CIDX, PIDX, VICS, ebXML, UBL, SOAP, AS2, ebMS, cXML, UN/EDIFACT, UBL, X12, HL7 and OAG.

4.2.6 Oracle Business Rules (OBR)

OBR evaluates rules rapidly using a light-weight, high performance rules engine. It provides a platform to build decision services which are a type of business activity services. Decision services enable business rules to be invoked as web services from BPEL PM or other service orchestration frameworks. The decision service tooling in BPEL PM provides seamless integration between BPEL PM and OBR.

It is a best practice to decouple reusable business rules from Service implementations regardless of the service layer the Service belongs to. OBR provides the capabilities to achieve this.

4.2.7 Oracle WebLogic Server (OWLS)

OWLS is a J2EE application server that provides a robust and scalable platform to build and run SOA Services. As shown in Figure 4–2, OWLS can be used to build business activity, data and connectivity services using standards based technologies like J2EE, Web Services, EJB, and JDBC. OWLS is omnipresent in the service platform as it is the underlying container for most of the products shown in the diagram. While it is possible to implement most types of Services from scratch using OWLS, other products might offer value-add on top of OWLS to increase productivity of Service development and to hide complexity underneath.

OWLS support a number of industry standards in the areas of Java, J2EE, Web Services, Security and communication. Appendix A provides a link to the full list of standards supported by OWLS.

4.2.8 Oracle Coherence (OCOH)

Coherence provides in-memory data grid capability to the Data Services layer. It provides replicated and distributed (partitioned) data management and caching services on top of a reliable, highly scalable peer-to-peer clustering protocol. Although Coherence is not a Service implementation tool, it adds data caching, failover and high availability capabilities by means of the data grid and play a key role in improving Service performance.

4.2.9 Oracle Data Services Integrator (ODSI)

ODSI provides the ability to quickly develop and manage federated, bidirectional (read and write) data services for accessing single views of disparate information in a standards based, declarative manner.

ODSI provides a wide array of data services designed to improve data access from disparate data sources for a wide range of clients. Using such services, organizations can create virtual databases that can more quickly and accurately model their dynamic enterprise.

Data services encapsulate the logic for reading, writing, creating, updating, and deleting information, insulating data consumers from having to contend with multiple data source formats and connection mechanisms.

ODSI supports the latest Web Services standards that include SOAP, UDDI and a slew of XML based standards including XQuery 1.0 and XPath 2.0.

4.2.10 Oracle Data Integrator (ODI)

ODI is a next-generation Extract Load and Transform (E-LT) technology that offers the productivity of a declarative design approach, as well as the benefits of a platform for seamless batch and real-time integration.

Data Integration provides the important aspects for moving data and turning it into re-usable information – it acts as a key enabler for each architecture principle (e.g SOA, BI, or Data Warehousing) and can also act as a key bridge between them. ODI provides high-volume, high-performance bulk data movement and transformation capabilities to the Service platform and exposes rapid data integration as Services that can easily be consumed for process orchestration.

ODI's standards support includes: Web Services standards, LDAP, SQL, JDBC and ODBC.

4.2.11 Oracle Complex Event Processing (OCEP)

OCEP provides a rich, declarative environment for developing event processing applications to improve the effectiveness of your business operations. Oracle CEP can process multiple event streams to detect patterns and trends at real time and provide enterprises the necessary visibility via Oracle Business Activity Monitoring (Oracle BAM) to capitalize on emerging opportunities or mitigate developing risks.

OCEP is a key element of Event Driven SOA (EDSOA) and provides inbound and outbound connectivity services. OCEP includes out of the box adapters for JMS and HTTP connectivity. OCEP Adapter SDK provides tools to create custom adapters that listen to incoming data feeds. This allows custom connectivity services to be developed and deployed.

OCEP is discussed at great length in the ITSO EDA documentation.

4.2.12 Oracle Integration Adapters (OIA)

OIA uses standards based on the J2EE platform to create a service orientated approach to unlocking the assets that have evolved in IT environments. OIA provides adapters to expose connectivity services to a number of backend systems including packaged applications, mainframes, legacy systems, technologies and protocols. OIA includes a lightweight adapter SDK that enables any JCA-compliant adapter to be rapidly integrated with the SOA platform.

OIA is fully standards-based ond is compliant with both the J2EE Connector Architecture (JCA) and Web Services Architecture. OIA standards support includes JCA 1.5, XML, WSDL and WSIF.

4.2.13 Oracle Web Center Interaction (OWCI)

OWCI is an integrated, comprehensive collection of components used to create enterprise portals, collaborative communities, and composite and social applications. WebCenter Interaction provides capabilities for social computing and Web 2.0 initiatives within the enterprise.

OWCI conforms to JSR-168 portlet standard and supports **Web Service Remote Portlet** (WSRP) standard for building consistent reusable presentation services.

4.2.14 Oracle WebLogic Portal (OWLP)

OWLP is the best-of-breed portal framework for creating highly interactive composite applications in a SOA environment with a powerful, integrated set of design-time

tools for Java developers and strong support for standards. This framework includes a rich, graphical environment for developing presentation services, portals as well as browser-based assembly tools for business experts. WebLogic Portal simplifies the production and management of custom-fit portals, allowing you to leverage a shared services environment to roll out changes with minimal complexity and effort.

OWLP supports **JSR 168** portable portlets, JSR 170 repository and Web Service Remote Portlets (WSRP) federated portals. In addition to these, it also supports the following Java standards: JSF 1.2, Struts 1.2, Beehive 1.0.1, JSP 2.0 and JSTL 1.2

4.2.15 Oracle Portal (PRTL)

Oracle Portal offers a declarative environment for building, deploying, and maintaining world-class enterprise portals. For employees, partners and customers who want to access business applications, content, collaborative tools and web sites, Oracle Portal brings a complete solution that is tightly integrated with Oracle Application Server. It provides the ability to build presentation services by means of standards based portlets.

Oracle Portal supports WSRP for remote portlets and **Java Portlet Specification** (JPS) that includes JSR 168 and JSR 286 for interoperable portlets.

4.2.16 Oracle Universal Content Management (OUCM)

OUCM is a unified enterprise content management platform that enables document management, web content management, digital asset management, and records retention functionality to build the business applications. OUCM includes Oracle Content Portlet Suite that implements a number of interaction portlets such as Search portlet, Library portlet and Workflow queue portlet. Content management and content presentation functions can be implemented as reusable presentation services that can be composed into standards based portals. OUCM also provides the ability to expose any content/collaboration manager functions as Services.

5

SOA Infrastructure Deployment View

This section describes the deployment view of the SOA infrastructure.

SOA infrastructure topology defines the architecture tiers and how various infrastructure products are integrated in a typical deployment.

The deployment model describes how SOA infrastructure components are deployed.

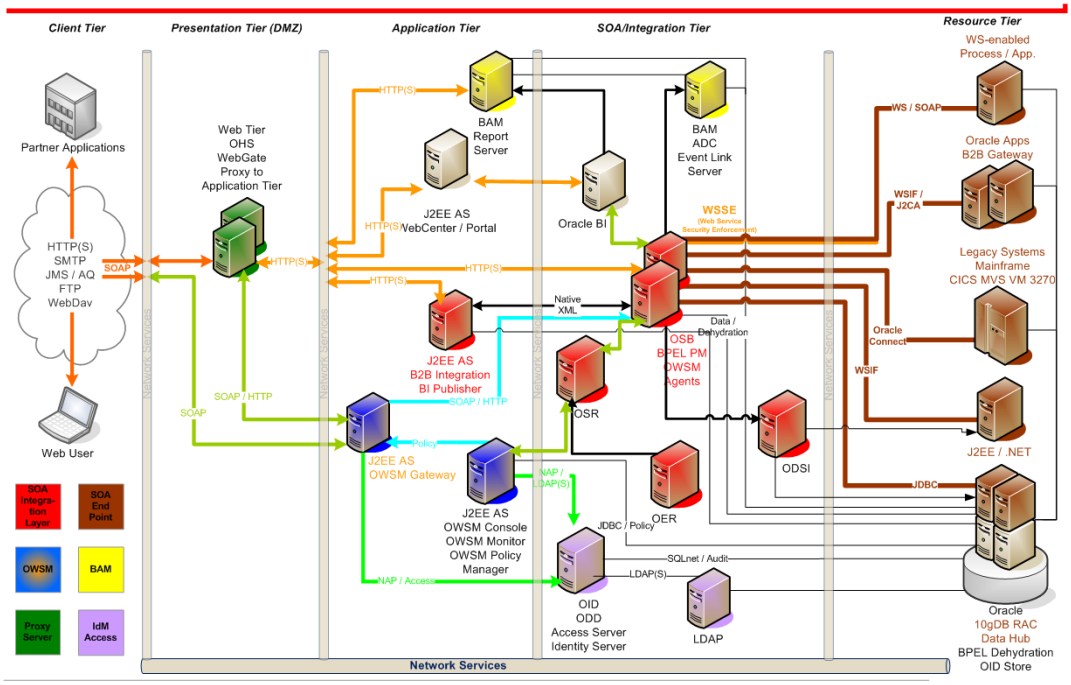
Product deployment strategies and the nuances of deploying the individual products are discussed as well.

This section ends with a discussion of the infrastructure best practices, architecture trade-offs and federated SOA topologies.

5.1 SOA Infrastructure Topology

The broad portfolio of Oracle middleware products offer the capabilities required to build a comprehensive SOA infrastructure. The pattern of evolution of this infrastructure might vary depending on the goals and SOA maturity of the organization but nevertheless, an optimized SOA infrastructure will require the components depicted in Figure 5–1. It shows the implementation of the SOA infrastructure using the Oracle SOA Fusion Middleware products organized by common tiers. The diagram also shows other related components like BAM, BPEL and Security that will be covered in other documents.

Figure 5–1 Oracle SOA Infrastructure Topology (Sample)

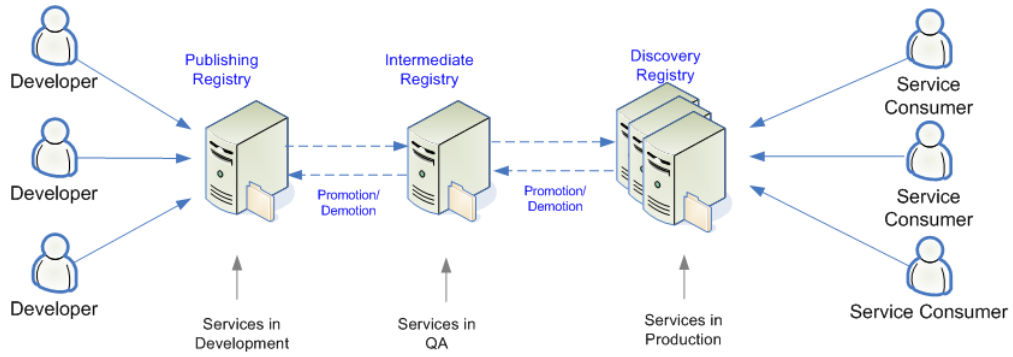


The client tier shows the external clients accessing the services in the trusted network through a DMZ. There are potentially internal consumers within the organization consuming the Services. OWSM client agents are configured on the client side to achieve policy enforcement on the client side and message encryption. OWSM gateways manage services from multiple providers and are deployed independently to control access and meter services. OWSM agents are deployed on the service bus or on the provider containers to implement last mile security.

The OSB service bus is deployed in a clustered environment and is a central element of the SOA infrastructure. It can be scaled horizontally or can be deployed in more sophisticated federated fashion, which is discussed in Section 5.1.1. The OSB and the registry are integrated so that the bus can import the services and subscribe for notifications. OSB can also publish the services directly to the registry although it is discouraged from a best practice perspective. The best practice is to promote services through the registry chain with the service promotion process. OWSM can also be integrated with the registry for automatic discovery and registration of services.

Oracle Enterprise Repository is shown in the diagram pushing the service metadata to the registry. This generally happens in the development environment publication registry. The testing environment has an intermediate registry which gets the services through the promotion workflow process. The production environment registry is a discovery registry. The intermediate registry pushes the service metadata to the discovery registry through the approval process. This process is shown in Figure 5–2.

Figure 5–2 Service Promotion

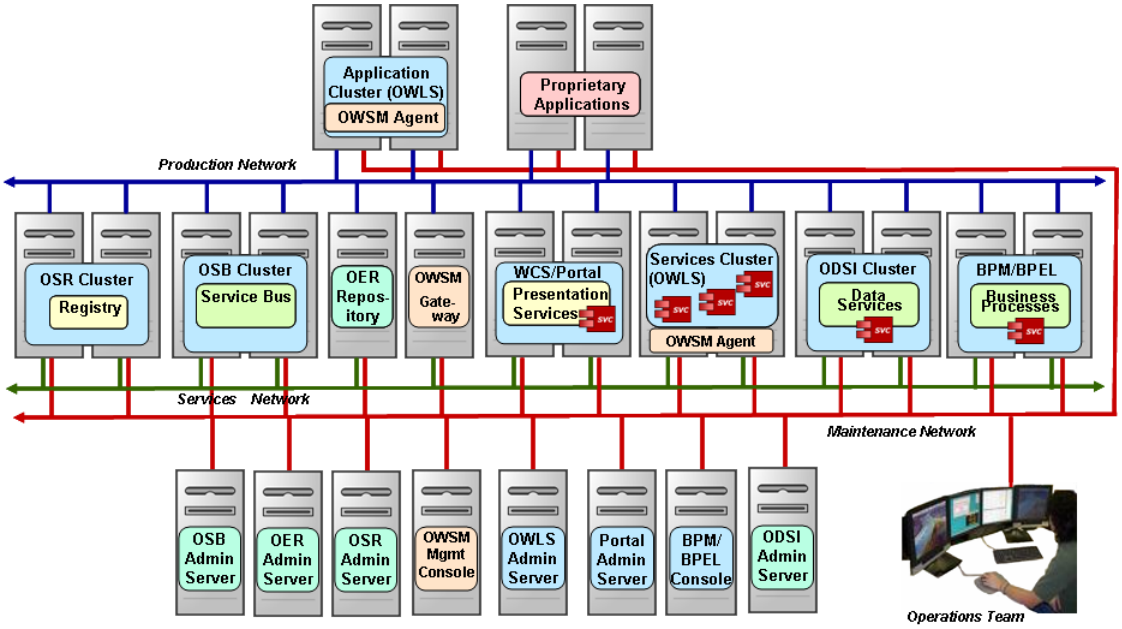


Although not considered part of the core SOA infrastructure, some of the products shown in the diagram provide a platform to implement the Services. BPEL PM, OBPM, WCS and ODSI are some of the examples.

For simplicity, this diagram shows only the key interactions in the SOA infrastructure and not all interactions. The administration servers are not shown explicitly in this implementation diagram but depending on the size and complexity of the deployment multiple domains and multiple administration servers might be required.

A number of factors influence the way SOA is physically deployed in an enterprise, including the service deployment and ownership model, availability of shared infrastructure, the way services are defined and used, availability and scalability concerns, security concerns, private and public services, the number of organizations involved, and the number of data centers involved. Each of these factors helps to determine how services, servers, and SOA elements, such as ESBs and registries should be deployed. A sample deployment model for a single-segment SOA Infrastructure system is depicted in Figure 5–3:

Figure 5–3 SOA Infrastructure Deployment Model



The sample approach illustrates the positioning of infrastructure elements across three separate networks. The production network is a protected network that connects applications running in the production environment. This is an existing network that may actually be made up of several physical networks linked together. Existing applications can access UDDI registry services for dynamic binding and service discovery. They can also access the Service Bus, which hosts proxy services that provide indirect mediated access to Services in the SOA environment.

A Services Network is shown connecting elements of the SOA environment. This network provides connectivity between the Service Bus and Services in the services cluster. It also links to the registry and security services. The reason it is separated from the production network is to eliminate the possibility to access Services without going through the Service Bus. Other methods include access control mechanisms, such as id and password protection, digital certificates, mutual authentication, etc. These can also be used by the services to restrict access to the Service Bus, thereby eliminating direct access by service consumers.

The third network is a maintenance network which is used by the Operations Team to administer various infrastructure components. This network connects to all systems in the production environment as well as all maintenance systems. It allows administration servers and operations terminals access to control production applications, services, and infrastructure.

5.1.1 Product Deployment Strategies

Each Oracle infrastructure product has its own unique deployment options and considerations. Most are deployed to applications clusters, such as Oracle WebLogic Server (OWLS), either in the form of applications running in the server environment or components of other applications. The following sections provide a brief look at each product. Detailed installation and configuration information is available online. It is recommended that online documentation is used to architect any solution as it is far more descriptive than this document is intended to be, and it is updated with each new product version.

5.1.1.1 Oracle Service Bus (OSB) Deployment

Scalability of Services Bus is achieved by clustering the instances. Oracle Service Bus (OSB) runs within a application server cluster. Clustering allows OSB to run on a group of servers that can be managed as a single unit. In a clustered environment, multiple machines share the processing load. OSB provides load balancing so that resource requests are distributed proportionately across all machines. An OSB deployment can use clustering and load balancing to improve scalability by distributing the workload across nodes.

The OSB console is used to design and compose the service wiring between consumers and producers. It is also used to set up the management and monitoring capabilities that will be enforced at runtime. These capabilities are then propagated to the runtime environment for execution. The OSB admin server, which hosts the console, is not required to be active in order for the runtime services of the service bus to function correctly.

5.1.1.2 Oracle Enterprise Repository (OER) Deployment

Generally only one instance of the repository is deployed with an optional additional instance for training and testing purposes. Since there should be one place where all the assets should be maintained with appropriate lifecycle statuses, one instance of repository should be maintained. The repository can be deployed on application servers like OWLS and can be clustered for high availability. Repository is not part of

the runtime infrastructure but it is a critical component of the design-time activities. So generally a single non-clustered instance would be sufficient for most use cases. OER provides browser based interface for managing and accessing the assets in the repository.

5.1.1.3 Oracle Service Registry (OSR) Deployment

The Oracle Service Registry (OSR), when utilized, can be deployed in a separate clustered environment, or within an existing OWLS cluster. The primary purpose of the OSR is to provide a runtime metadata registry for runtime discovery and invocation of services through the use of the UDDI interface. A web services interface is also available to the registry if needed.

OSR is shown deployed to its own OWLS cluster, which allows it to be managed independently from other applications, services, or infrastructure components. Management features are built into the runtime deployment, which means there is no additional remote administrative process deployed for OSR. Management is performed by accessing the production instances directly via a Web browser.

5.1.1.4 Oracle Web Services Manager (OWSM) Deployment

Oracle Web Services Manager (OWSM) is comprised of a number of management components that run on production management systems, as well as remote agent components that are co-located with service providers and consumers. The management capabilities can be divided into categories such as: service level monitoring, logging, and exception handling. Each can be deployed to different systems, as performance and load characteristics warrant. Management components may be deployed to OWLS clusters. OWSM agents are available in two varieties: gateways and agents. Gateways are deployed standalone whereas the agents are codeployed with the endpoints.

Please note that the Deployment Architecture diagram depicts an OWSM agent deployed with one of the application clusters, but not with the Services. The purpose is to show OWSM as a client side agent that measures round trip performance for a particular client. In this scenario it is not represented as the provider of security or mediation.

Since proxy services are considered the true service interface, it naturally follows that the service provider endpoint is the Service Bus. Logic behind this interface, including the interface to back end business services, are considered part of the service implementation. Therefore it also follows that if OWSM were to be deployed at each service endpoint, it would be deployed with OSB, not with the Services in the services cluster.

5.2 SOA Infrastructure Best Practices

This section describes best practices with respect to SOA infrastructure.

5.2.1 OSB Architecture Trade-offs

OSB provides a very flexible, feature-rich Service Bus. Its role as an active intermediary requires it to be positioned between the service consumer and provider.

5.2.1.1 Bus Architecture

The approach taken by OSB is somewhat "hub-based" in that it is positioned in the middle of many consumers and providers. A bus architecture is often drawn as a pipe that messages move through. This is somewhat misleading in that the pipe is

essentially the network between components and OSB in another endpoint on the network. To be more specific, OSB is a midpoint on the network with respect to service providers and consumers.

5.2.1.2 Endpoint Deployment

Another option in terms of Service Bus architecture is an endpoint deployment. This involves the installation of Service Bus components at every endpoint, rather than one component at the midpoint. Endpoint components, usually called agents, are synchronized in such a way that they work in harmony, following a common configuration. The benefit of this approach is there is only one network hop. Usually the real service provider's endpoint is advertised since the agent intercepts inbound and outbound traffic as part of a filter or handler. It also provides the ability to measure performance and provide security controls at the endpoints where it is most useful.

The downside of endpoint deployments is primarily two-fold. First, it can become a management nightmare since there can be hundreds of endpoints and many different technologies involved. The likelihood is high that some platform or service interface technology in use will not be supported. Second, lightweight endpoints generally do not provide features such as content-based routing, message transformation, and service composition. For these reasons the midpoint solution OSB provides is recommended.

5.2.1.3 Service Bus Appliance

Another option to OSB is a service bus appliance, or firmware-based solution. These may provide greater throughput at the cost of limited functionality and flexibility. Their downside is that they are usually less flexible, provide fewer features, and are harder to configure. One must consider these trade-offs when evaluating Service Bus components.

5.2.1.4 Message Bus

A final option to consider is the use of Message Bus products as a Service Bus. There has been somewhat of a convergence of products in this regard, with Service Bus features being added to Message Bus products. Rather than provide a point-by-point product comparison in this document that will quickly become dated, it would be better to simply recommend choosing the right tool for the job. A Service Bus provides many critical features and supports standard integration protocols. Message Bus products tend to use proprietary integration techniques and cost more. Features, price, and standards support help drive the need for pure-play Service Bus products, such as OSB.

5.2.2 Mediation vs. Orchestration

Another convergence taking place is in the orchestration of activities with respect to business services and business processes. OSB provides the ability to design process flows that control a series of activities to fulfill service requests. The process flows are relatively simple, and are meant to support the modeling and orchestration of technical operations pertaining to the execution of a service. The intent of OSB is to act as a proxy, or mediator for service providers. Though the proxy services are represented as a flow of activities, the service logic itself belongs to the service provider. In addition to invoking the service provider, OSB flows may include activities to support logging, routing, message transformation, and security. These activities generally fall under the category of common infrastructure, or foundational activities (as opposed to business logic / decision making activities).

More sophisticated orchestration is available with other products such as Oracle BPEL Process Manager and Oracle BPM. These products are designed for orchestration, and can expose orchestrations as services. It is easy to see where they may be thought of as being interchangeable. OSB, BPEL PM, and OBPM engines all provide some form of process or flow modeling with exposure as services.

The difference between OSB and the BPM products is that OSB is designed as a configuration-driven infrastructure component. This allows it to be used by multiple project teams, supporting many applications and services, without being tied to any project's delivery cycle. Though it supports custom coding, it does not require any such coding or deployments in order to add, change, or remove proxy services. This makes it ideal as a mediator in the form of a common infrastructure component. It provides the mediation capabilities that form an intelligent, loose coupling between service consumers and providers. It is configured by Operations staff or developers, and ideally contains no business logic.

OBPM and BPEL PM orchestrations are generally used as service provider platforms. They support the modeling of business or technical orchestrations that can be exposed as services. BPEL PM is well suited for technical orchestrations. An example is data synchronization, where complex data update scenarios can be modeled and executed.

Orchestration models may be represented using an XML notation. Common notations include BPEL and **XPDL**. The intent of standardizing this notation is to allow tools and products from multiple vendors to interact, and to eliminate vendor lock-in.

Regardless of the standard chosen, the focus of products aligned with these standards is on business process modeling and execution. Business processes are created by Business Analysts to define the flow of activities for business functions. The process models are executed by BPM tools. Processes (and sub-processes) may be exposed as services in a SOA. Like other types of services, process-based services should also be mediated. Therefore, the recommended approach is to use OBPM, BPEL, **BPMN**, and other process modeling tools as service providers, and use OSB as the mediator. This supports a beneficial separation of concerns, where each tool is used by its target audience for its intended purpose. Business logic resides in the business process modeling tool, readily available to the Business Analysts, while mediation activities reside in the Service Bus, which is supported and configured by Operations and/or development personnel.

5.2.3 OWSM Architecture Trade-offs

The OWSM distributed architecture stands in stark contrast to the centralized nature of OSB. Both provide similar features, such as security enforcement, routing, message transformation, and the ability to monitor performance and usage, and log access to services.

A key benefit of OWSM (with respect to these common features) is its ability to instrument the client or deploy Client Agents. This allows it to measure response time from the client's perspective rather than from a point in the middle. It also has the ability to scale out and yet offer centralized management.

OSB clusters follow a more silo'ed approach where each cluster is maintained individually. Each cluster must be updated when services are provisioned or changed. In addition, OWSM offers management capabilities that other Oracle products (including OSB) do not offer, such as message-level security policy management, SLA management, and run time service and dependency detection.

The downside of adopting such a distributed architecture is the need to install it at every endpoint. This can create management challenges when it comes to maintaining these deployments as the number of endpoints escalates. It is also prone to technology

limitations. For instance, it is unlikely that agents will be available for every platform that hosts services, and even less likely for every type of client (service consumer). This opens up the possibility that holes will exist in the SOA environment where OWSM can't be applied. For these reasons, OWSM is positioned as providing essential capabilities of Monitoring, Management, and Security Frameworks.

5.3 Federated SOA Domains

During the initial stages of SOA maturity, as with most IT initiatives, SOA initiatives seem to work from the bottom up. Groups (lines of business, portfolios, divisions, etc.) begin to experiment with SOA, and embrace it independently from each other. They deploy Services and SOA Infrastructure within the boundaries of their organization. And often they are responsible for maintaining their own assets. This leads to the notion of SOA domains.

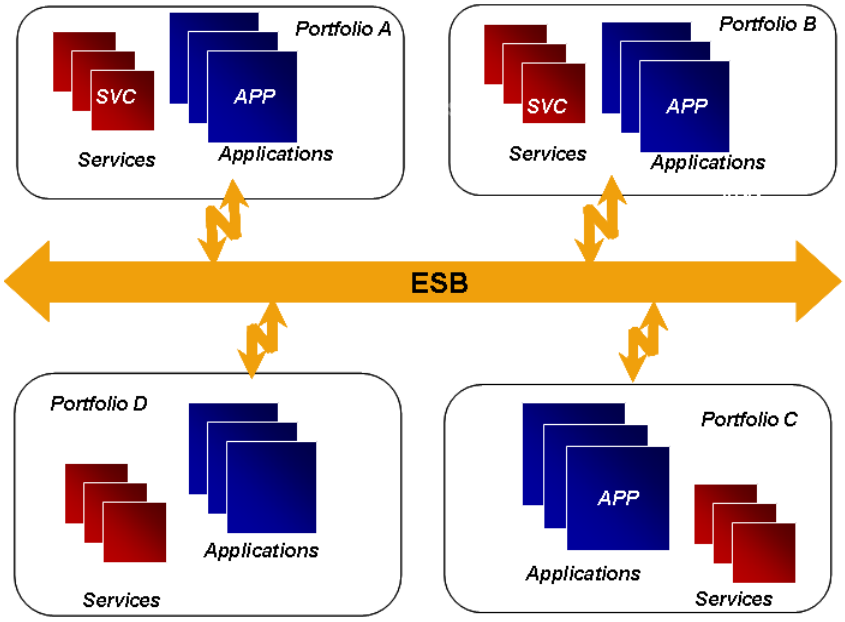
The boundaries of a SOA domain generally fall along major organizational boundaries. The reason is based on funding and administrative responsibility. As organizations are able to fund SOA initiatives, SOA will grow and evolve. Funding for infrastructure is often absorbed at the portfolio or project level. This promotes a more localized SOA domain, different than if funds were allocated for infrastructure at the corporate level.

Organizations must also be willing to trust that services and infrastructure they depend on will be available. They need assurances that reliability will not be impacted. Maintaining authority over, and responsibility for, computing resources is one way to achieve a level of comfort with regard to system reliability. For this reason SOA domains may align somewhat with operational boundaries, i.e., each organization that has an operations team will establish and maintain its own SOA domain.

The natural occurrence of multiple SOA domains within a company does not need to be a detriment to enterprise SOA. It is a result of increasing SOA maturity within the organization. It is important to view these domains as federated enterprise-wide deployments rather than multiple silo'ed departments. This requires the establishment of an overarching Reference Architecture that dictates such things as interoperability standards, service discovery techniques, and service usage guidelines. With this in place, the company can have the best of both worlds: localized control and enterprise-wide SOA.

There are many patterns to support federated infrastructure deployments. The way to determine how it should work for a particular company should begin with the service engineering process. More specifically, who is responsible for developing and maintaining Services. If a centralized "**Service Factory**" model is used, then services and infrastructure may be needed at the enterprise level. Each organization within the enterprise may optionally deploy services and infrastructure to support services that have a local scope, or fall outside the purview of the Service Factory. An example of this is illustrated in Figure 5–4.

Figure 5–4 SOA Domain Federation



5.3.1 Federation Topologies

Depending on the needs of an organization, three different topologies can be considered: distributed, centralized, or a hybrid of the two. A distributed topology offers maximum autonomy for individual departmental Service Buses. However, this topology reduces enforcement of central policies. Centralized topology reverses the pros and cons of the distributed approach.

Federated Service Bus topology is an approach for service bus deployment consisting of number of service bus domains that are deployed across the enterprise SOA Infrastructure. Note that a Federated Service Bus topology does not imply a certain type of relationship between the buses deployed, nor a particular way of provisioning and configuration. However it implies that the service busses deployed across the enterprise have connections to each other to provide virtual single service bus functionality.

There are multiple types of federated service bus topology:

- Centralized Federation Topology (Hierarchical Service Bus hub connections)
- Distributed Federation Topology (Autonomous Service Bus hub connections)
- Hybrid Federation Topology (Both hierarchical and peer-to-peer hub connections)

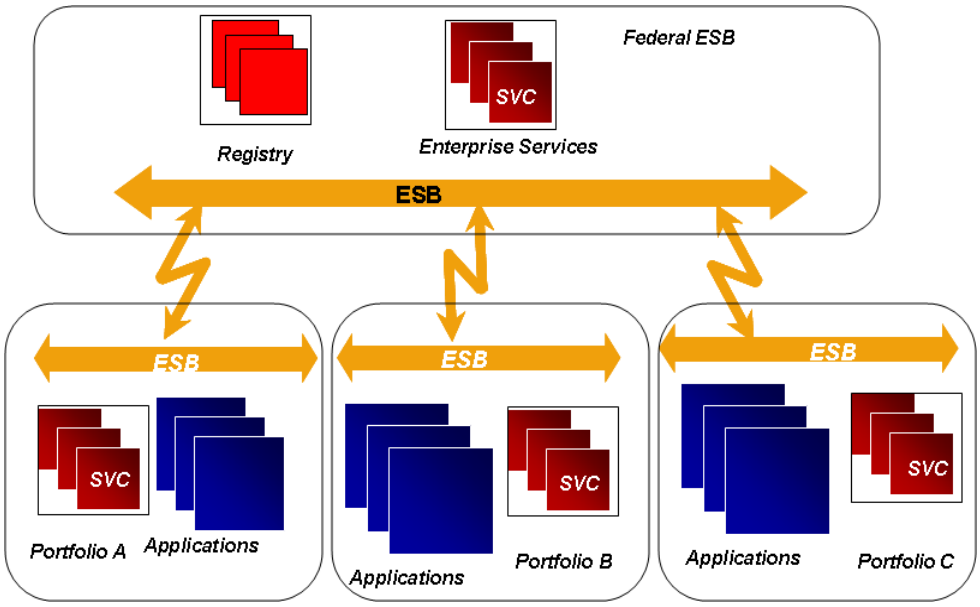
5.3.2 Centralized Federation Topology

Centralized Federation Service Bus Topology is a specific setup in which the Service Buses are deployed using a well defined hierarchical relationship. For example, they may be configured with a master or a central supervisor service bus and a number of subordinate service busses. The master service bus usually serves requests from subordinate service busses.

The principle responsibility of the master service bus is to provide routing, and aggregation of messages to and from multiple service providers, and transformation. The main service bus also acts as the main security gateway for the incoming external services.

The federated or subordinate service buses are usually scoped to one or more departments that are individually or collectively responsible for providing services using that bus. The rationale behind such a hierarchy is to allow individual organizational units to be able to effectively control all aspects of the services that they provide or are otherwise responsible to provide.

Figure 5–5 Centralized Federation Topology



This topology requires all the divisional or federated service buses to route the requests to one another via the central backbone service bus.

Figure 5–5 depicts multiple SOA Domains, one at the enterprise level and two at the portfolio level. Services offered at the enterprise level are owned and managed by a corporate Service Factory team. They are made available via the ESB associated with that domain.

Each portfolio can host its own local services. These are only available within the portfolio, otherwise they are promoted to the enterprise level. Applications within the local portfolio domains can leverage local services as well as enterprise services. Each domain contains a registry to advertise services within that domain. The registry at the enterprise level is available to all domains, whereas the registry within the portfolio domains is private to that domain.

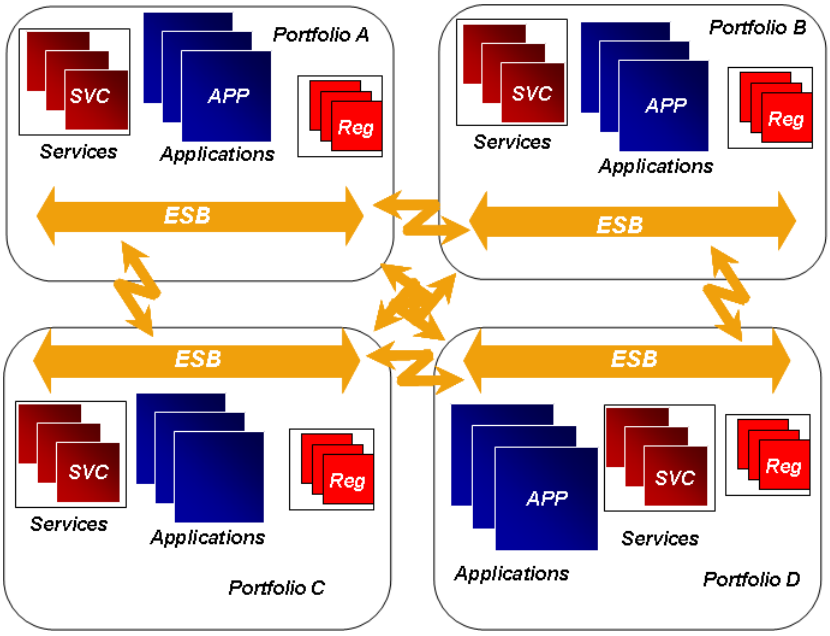
This example requires the configuration of multiple ESBs. There are basically two ways to configure them: as gateways to and from the domain, or as endpoints directly exposed to service consumers of adjacent domains. In this example the arrows indicate a gateway configuration. This means that traffic from one domain to another must pass through two ESBs - the local ESB and the remote ESB. Proxy services must be configured in the local (portfolio) domain in order to access enterprise services. This adds management overhead, but in return offers the ability to mediate messages, transports, and security, between domains. Other deployment variations may include single instances of the ESB and/or registry.

5.3.3 Distributed Federation Topology

Distributed Federation Service Bus Topology, consists of multiple autonomous service buses communicating with each other directly as peers. Each service bus hosts a

number of services and accepts requests from a number of clients. There is no central hub that all requests pass through.

Figure 5–6 Distributed Federation Topology

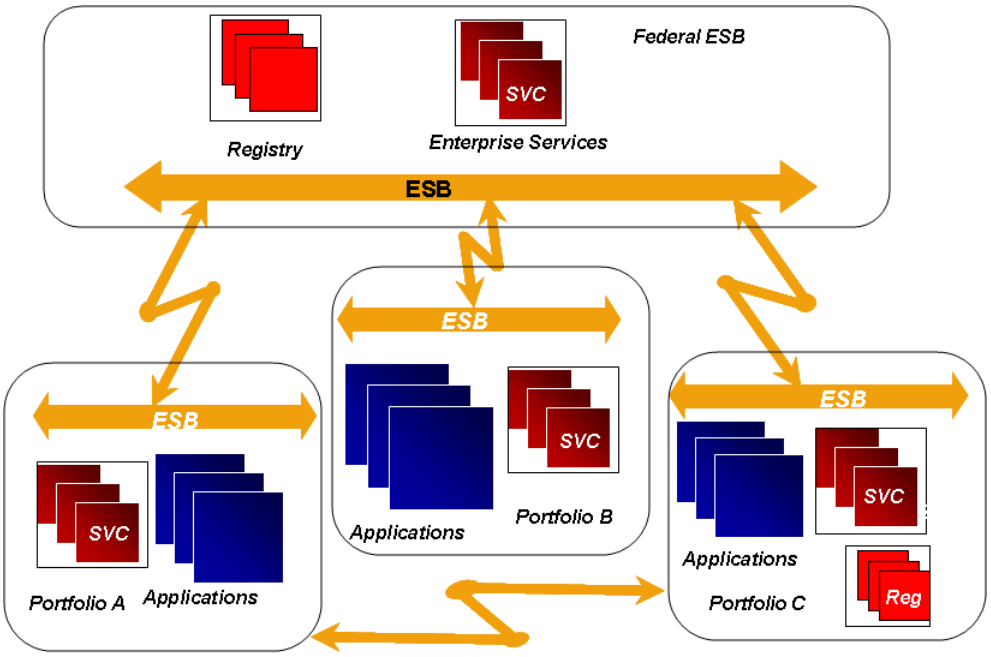


The distributed topology for federation involves direct service invocations across different divisions.

5.3.4 Hybrid Federation Topology

Hybrid Federation Service Bus Topology is one that utilizes both central and distributed service bus topologies: Although there is a central bus in the topology, bus-to-bus communication is also allowed under certain circumstances. There can be large variations of this topology since there many topology options between centralized and distributed.

Figure 5–7 Hybrid Federation Topology



5.3.5 Drivers for Federated Service Bus Configurations

There are two main categories of drivers for distributed service buses. The first category of drivers is related to the needs of the large organizations to manage different parts of the organizations in a federated manner involving separation of the responsibilities and policies among various organizational units. The second category is technical which has some similarities to the drivers for having multiple web servers.

While Centralized federation of the Service Buses can result in a number of the benefits described above, centralization can also incur some costs.

- Some federation configurations may create one or more additional network hops which may impact performance. Care must be taken to not take the federation to extreme limits.
- A large number of Service Bus instances can create their own management problems. Hence, for a small number of services, homogeneous buses (such as single protocol, having similar QoS and SLAs), or buses having similar security and access policies, federation of the Service Buses should be considered carefully.

6

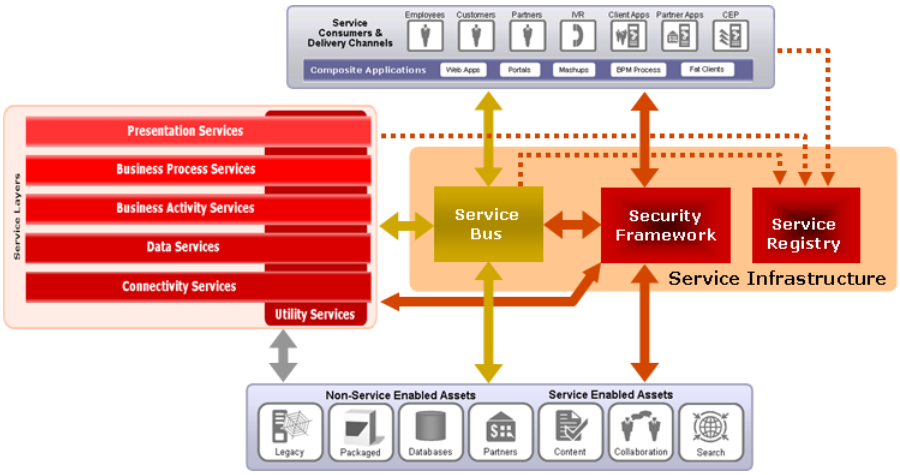
SOA Infrastructure Process View

The Process View describes the runtime interactions of the various components of the SOA infrastructure. In this section, we will focus on the runtime interactions between various logical components that make up the SOA infrastructure.

6.1 Runtime SOA Infrastructure Process

At runtime, the SOA Infrastructure acts upon the configuration constructed at design-time. Primarily this includes the service mediation functions performed by the Service Bus, the security functions, such as authentication, authorization, auditing, encryption, and policy enforcement capabilities of the Security Framework, and optionally, the runtime lookup capabilities of the Registry. To simplify the diagram, monitoring and management has been separated into a diagram of it own, which is provided in Section 6–2.

Figure 6–1 Runtime SOA Infrastructure



The SOA Infrastructure facilitates service interactions at runtime. The Service Bus is used to establish a loose coupling between service consumers and service providers. In this diagram, service consumers are at the top and include various types of end users and applications that might consume Services. Service providers come in two forms: the Services themselves and IT assets that expose functions and data directly to the Service Bus as services. Interactions between consumers and producers are shown by the arrows running across the Service Bus. Note that Services may depend on existing IT assets in order to perform their functions. A direct link between them is shown in

gray. This link would include integration embedded within the shared service, (out of band and not exposed for direct consumption by others).

The Security Framework extends traditional security architecture to include capabilities required by SOA. This includes policy-based security, either in the form of access control policies, or in the form of message level security policies. Access control policies are not necessarily tied to SOA, since they pertain equally well to traditional architectures. However, they become more applicable as the need for interoperability increases. Policies provide a means to unify standalone application silos. In a SOA environment they can control access to services based on roles, groups, or even individual users.

Message level security provides a means to secure messages end-to-end. In a highly distributed architecture such as SOA, this enables identity propagation, confidentiality, and integrity across multiple endpoints while data is being sent, processed, and persisted.

The Security Framework stores both types of policy information and is used to enforce policy as well. Since enforcement happens in many places, the framework extends into the computing environment, often handled by application containers. Therefore, the links between the Security Framework and all other runtime components may be implemented as a combination of low-level container APIs, as well as higher-level service interfaces. Low-level APIs to facilitate faster response times and local decision making, and higher-level service interfaces to provide reusable security as a service.

Lastly, if the Registry is used to store runtime metadata, then it will be queried by service consumers as part of the service invocation process. The Registry provides UDDI functions to obtain access to runtime policies. The dotted lines represent UDDI lookups by consumers, services, and the Service Bus.

Note that components of this architecture diagram are intended to represent logical entities rather than physical components. Typically one or more products will be deployed to implement the logical components.

6.2 SOA Management Infrastructure Process

As the level of SOA commitment increases, monitoring and management become more important. In fact, higher levels of SOA maturity require organizations to understand what is transpiring in their SOA environment. They must be able to monitor applications, services, and infrastructure, and react to situations at hand. The greater ability to monitor and react, or manage, the greater value SOA will have to business and IT.

Figure 6–2 Management Infrastructure Process

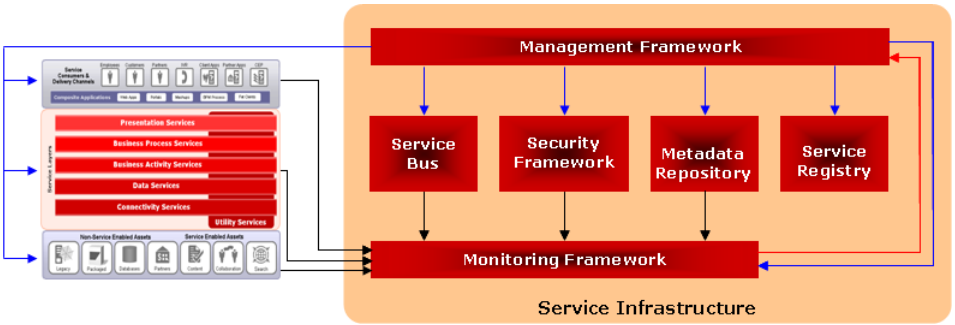


Figure 6–2 illustrates monitoring and management as three types of interactions. First, IT assets including Services, Service consumer applications, and SOA Infrastructure components are instrumented to report on various events and conditions within the SOA environment. The types of information they log can vary widely depending on the types of events in an organization. Examples events include:

- Service invocations;
- Service exceptions;
- Significant business events;
- Business process exceptions;
- Performance and load characteristics;
- Service and resource availability;
- Security alerts;
- Service lifecycle transitions;
- New service availability;
- Service deprecation notifications.

Decisions regarding which events to capture should depend on business and IT objectives. The point of monitoring is to provide a means to measure effectiveness toward business and IT goals. Therefore, the process should start with the examination of these goals and the determination of which data points, or **Key Performance Indicator**s (KPIs) are needed to evaluate effectiveness. Additional points may be included to support variations and extensions of the initial evaluations.

The Monitoring Framework provides a means to collect information and present it to the Operations Team or others responsible for data evaluation. The framework may consist of a number of components including traditional operational monitoring products like Oracle Enterprise Manager. It may also consist of products designed to monitor service usage, interactions, and dependencies, such as OWSM.

Once data is collected, it needs to be made available for evaluation. Data may be presented in the form of indicators on a management dashboard, or formatted into reports. This flow of data from the Monitoring Framework to the Management Framework is represented by a red arrow. In addition to making data available, the monitoring framework may feed data directly into management processes. These processes would automatically react to events in the system and trigger management responses. This is truly the end goal in terms of SOA optimization - to continuously monitor the computing environment and adjust to events and trends in real time.

The arrows leaving the Management Framework illustrate its influences on Services, consumers, IT assets, and the SOA Infrastructure. This framework will likely be implemented as a collection of management interfaces and tools. Some interfaces will be administrative user interfaces that are purchased with other infrastructure products. OSB, OWSM and OEM are examples of this, where service management and policy management interfaces are included with the mediation and security products. Other management interfaces may include those that accompany virtualization software, BPM tools, and rules engines. Since every environment will be unique, it is not realistic to provide specific reference architecture recommendations. Typical management concerns include:

- Managing actual service usage (vs. Service Agreements)
- Managing service availability (vs. SLA commitments)
- Managing service dependencies (vs. dependency models)

- Managing response time and load
- Managing service versions and the routing of consumers to versions
- Managing security policies and access control
- Managing the service lifecycle (service candidate approvals, versions, etc.)
- Managing transformations, routing, and mediation
- Managing service discovery and taxonomy

7

SOA Infrastructure Development View

The development architecture focuses on the actual software module organization on the software development environment. Design-time activities include requirements analysis, architecture analysis, system design, implementation, deployment, and provisioning. They are essentially the same activities that have been done for many years as part of the software development lifecycle, however, some have changed with the adoption of SOA.

7.1 Design-time Process

The software development process has been augmented for SOA to support the following activities as described in the Oracle Service Engineering Overview Practitioner's Guide:

- **Service Identification**: The recognition of functionality identified by a set of requirements that may constitute a service.
- **Service Discovery**: The act of locating pre-existing services, service candidates, or existing code that is similar to functionality needed by the current project.
- **Service Lifecycle Management**: The process of tracking services and service candidates through the development lifecycle and managing their status, progress, relationships to other services, ownership, versions, release plan, and dependencies.
- **Service Asset Management**: The act of managing assets such as contracts, schemas, **usage agreement**s, models, requirements, etc., that are associated with a service.
- **Service Provisioning**: The set of activities required to configure infrastructure in order to make a deployed service implementation useable in production.

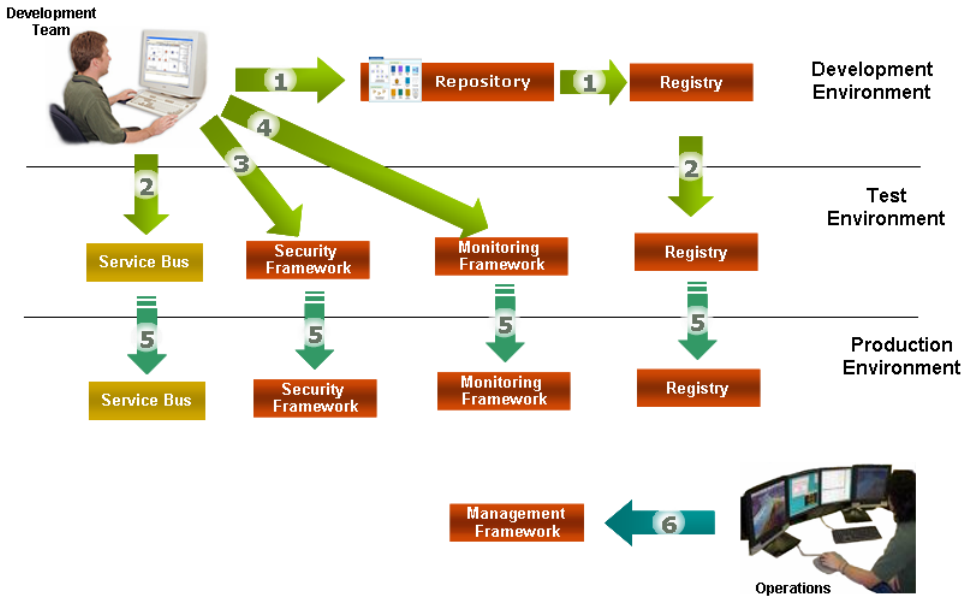
In addition to these service engineering activities, the following are also included:

- **Service Security**: The configuration of infrastructure necessary to provide for security of a service and the assets it uses.
- **Service Monitoring**: The configuration of infrastructure necessary to properly monitor a service in order to measure **Key Performance Indicator**s (KPIs).

The design-time process unfolds over a number of environments. It generally begins with a development environment that supports the requirements analysis, architecture analysis, system design, and implementation phases of the software lifecycle. It moves on to a number of test environments designed to support functional, integration, performance, quality assurance, and various final pre-production testing. The design-time process concludes as applications and services are deployed into production and provisioned for use.

Figure 7–1 illustrates this progression using a simple scenario with three environments.

Figure 7–1 Design-time Progression



1. In this scenario, the development environment includes a repository, which is used to support and manage the service lifecycle. The developers, service librarians, or persons charged with populating the repository, load service artifacts as they are created. They may also navigate and search the repository for existing services. By organizing requirements into related functional capabilities, and associating service candidates and services with the functional capabilities, it becomes possible to view a hierarchy of system functions and know which have been built as Services and which have not. It is also possible to search for functional characteristics to locate services. As a result, the repository supports service identification and discovery.

   Services within the repository are assigned a status, such as: In Progress, Available, Deprecated, and Retired. They are also assigned a version number so that multiple versions can co-exist and be managed appropriately. The assets, such as contracts, schemas, WSDLs, usage agreements, release plans, etc., are linked together so that potential consumers of the service can locate and download them. Through this linking, the repository supports asset management and dependency tracking.

   The repository is not intended to replace traditional code management tools. As code is developed, it will continue to be managed by configuration management tools. They provide the concurrency management, release management, and build capabilities for software assets. However, code artifacts related to services - specifically those needed by the consumer to integrate with services, may be copied into the repository in order to make the service discovery process easier. By grouping these assets with the services, consumers have an easier time locating and downloading them. They also have a greater level of assurance that they are getting the correct artifacts for a particular service and version.

   Services may also be entered into a Registry. The Registry may act as a public facing taxonomy of services, to be used by consumers that should not have access

to the Repository. It may contain enough artifacts to discover and invoke services, and only advertise services currently available for use. In this case the Registry would only need to be deployed in a public facing environment. However, if the Registry is used for runtime policy lookup or dynamic binding, then it would need to exist in the test environments as well as the production environment. Configuration additions and changes ideally would be propagated across environments using an export/import mechanism.

2. As services are developed, it becomes necessary to establish a proxy service for mediation and loose coupling. In this scenario the Service Bus is used, and the location is shown to be within the test environment. The Service Bus, Security Framework, and Monitoring Framework could also exist in the development environment, however it would not be essential, especially if these infrastructure components seldom require development activities. It is assumed for this scenario that infrastructure is primarily configuration-driven, and therefore isn't included in the development environment.

   In order to properly test the service, a proxy service is created in the Service Bus. It routes to the service implementation and becomes an extension of that implementation. It is here that some requirements, such as auditing, security, and interface-related (transport, protocol) requirements may be realized. Once the proxy is created, the proxy's interface becomes the service interface. Integration between the Service Bus and implementation code becomes a communication link within the service itself. The proxy interface and implementation interface could be exactly the same. An example would be if the proxy and service implementation logic were both exposed as Web Services. They might share the same WSDL, with the exception of the endpoint address. Regardless, the proxy interface is meant to represent the service's interface and should be the only interface exposed to service consumers.

   The service metadata is promoted from the development registry to the testing registry.

3. Services should specify security requirements within their Contract. These requirements may depend on infrastructure that is designed to provide security features, e.g., the Security Framework. In order to properly test security requirements, the Security Framework has been included in the test environment. It is here that security policies are configured and tested with the services. Policies may pertain to message level security, following the WS-SecurityPolicy specification, or something similar. They may also pertain to entitlements, or fine-grained access control. These policies generally follow the XACML specification, or an equivalent. Security policies can be custom written for a service, or may be reused across multiple services with similar security requirements.

4. Services may also be assigned monitoring requirements. These requirements may be common among all services, or all services of a particular type; or they may be unique to a particular service. For example, all services may be required to log an event when they are invoked, which may include the caller's identity and a timestamp. In addition, a specific service may be required to log an event when certain conditions occur, such as when a transaction over a certain dollar amount is performed. These requirements will need to be tested before the service is deployed into production. The Monitoring Framework may need to be configured to read such events and process them appropriately.

5. Once all testing cycles have completed, the service will need to be deployed into production and provisioned for use. The Operations team is generally responsible for the production environment and production deployments. In order to ensure

that the production configuration matches the final test configuration, infrastructure configuration will likely be migrated from the test environment to production, rather than manually re-entered in production. In this scenario the Service Bus, Security Framework, and Monitoring Framework configurations are exported from the test environment and imported into the production environment. The service metadata could be automatically promoted from the test environment to the production environment. Any final configurations or code deployments will also be handled by the Operations Team.

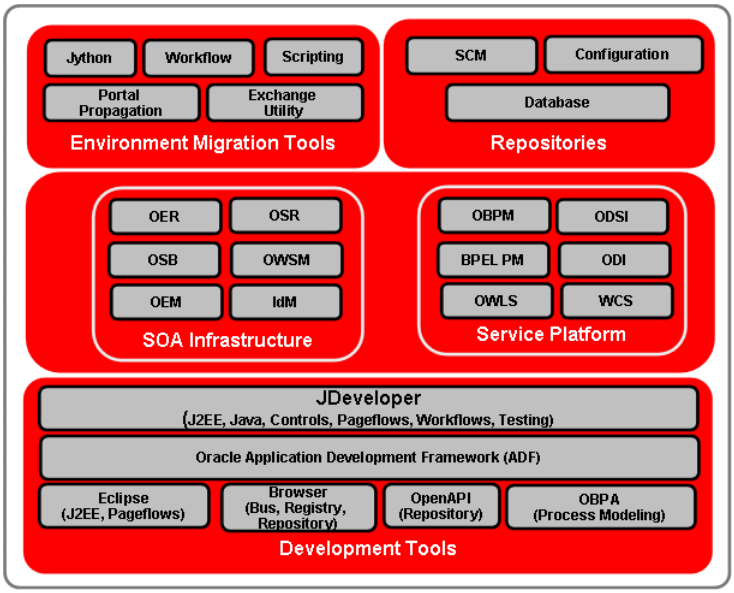
OSB and OSR have the capability to synchronize with each other. This mechanism could be used to populate proxy services into the production service registry. It can also be used in the reverse direction, to populate the Service Bus from the registry. This capability may be used to propagate services across organizational boundaries. For example, if each organization (department, portfolio, division) deploys its own infrastructure, then it may use the registry to obtain service configuration for services in other domains.

6. Finally, it may be necessary to adjust the service configuration as it applies to the production environment. This may be the case when load balancing or virtualization settings need to be made or changed based on intended service level agreements. These functions are performed via the Service Management Framework. Note that this framework may consist of a number of administrative interfaces to various pieces of the SOA Infrastructure. Over time this should congeal into a more uniform framework that establishes a consistent view of the overall system.

7.2 Oracle Fusion SOA Development Tools

Figure 7–2 shows the development architecture of the Oracle Fusion SOA stack. Oracle SOA infrastructure products are supported by development tools like Oracle JDeveloper, Eclipse and browser based tools. JDeveloper is the strategic unified development platform for all Oracle SOA products but for backward compatibility, Eclipse is also supported in some cases.

Figure 7–2 Oracle Fusion SOA Development Tools



Browser based tools are provided for development and administration related activities. It allows configuration driven development to be done using the browser. Oracle Business Process Analysis (OBPA) suite provides a component called Business Process Architect that can be used to design and model business processes. OBPM provides a Designer for business process modeling and simulation.

As developers develop the SOA assets, they are placed in various repositories that include **Source Code Management** systems, database repositories and configuration repositories. For example, the OSB artifacts are stored in the database but can be exported into external configuration files.

The middle layer shows Oracle SOA infrastructure products and Service implementation platforms. The products depicted in the diagram are listed by category below.

SOA Infrastructure:

- OER - Oracle Enterprise Repository
- OWSM - Oracle Web Services Manager
- OSR - Oracle Service Registry
- OSB - Oracle Service Bus
- OEM - Oracle Enterprise Manager
- IdM - Oracle Identify and Access Manager

Service Platform:

- BPEL PM - BPEL Process Manager
- ODSI - Oracle Data Services Integrator
- OBPM - Oracle BPM
- WCS - Web Center Suite
- OWLS - Oracle WebLogic Server
- ODI - Oracle Data Integrator

Developers have the flexibility of choosing from a variety of migration tools to automate the promotion of assets. These range from scripting approaches like Jython or UNIX shell scripting, to preconfigured promotion workflows. There are also specific tools like Portal promotion tool or Repository Exchange Utility for promoting specific components like portal configuration or service assets in the repository.

The Oracle Application Development Framework (Oracle ADF) is an end-to-end application framework that builds on Java Platform, Enterprise Edition (Java EE) standards and open-source technologies to simplify and accelerate implementing service-oriented applications. Oracle ADF simplifies the development of enterprise solutions that search, display, create, modify, and validate data using web, wireless, desktop, or web services interfaces. Used in tandem, Oracle JDeveloper and Oracle ADF give an environment that covers the full development lifecycle from design to deployment, with drag-and-drop data binding, visual UI design, and team development features built in.

8

Summary

SOA infrastructure plays a key role in accelerating adoption of SOA in an enterprise. It allows enterprises to jumpstart their SOA efforts by providing most of the SOA capabilities needed to build and deploy services. An understanding of SOA logical architecture and how it can be realized using technologies available in the market is an essential part of the SOA strategy. Oracle offers an end-to-end solution for the SOA infrastructure requirements and this document covered various views of SOA infrastructure and how best to leverage Oracle products to realize best in class and reliable SOA infrastructure.

SOA infrastructure should not be a simple installation of products, rather it should be the "realization" of the reference architecture which is aligned with the IT and business strategies. It is recommended to develop functionally driven SOA reference architecture and a multi-year SOA infrastructure roadmap and develop the infrastructure gradually. This prudent approach to building infrastructure not only ensures that the IT investment is expended with due diligence but also reduces the risks associated with rolling out SOA in the enterprise.

This document only covered the core components of the SOA infrastructure. Other complementary technologies, Service platform and related products are covered separately in the appropriate technology perspective documents.

A

Further Reading

The ***IT Strategies From Oracle*** series contains a number of documents that offer insight and guidance on many aspects of technology. In particular, the following documents pertaining to the SOA Infrastructure may be of interest:

A.1 Related Documents

ORA Integration - The ORA Integration document examines the most popular and widely used forms of integration, putting them into perspective with current trends made possible by SOA standards and technologies. It offers guidance on how to integrate systems in the Oracle Fusion environment, bringing together modern techniques and legacy assets.

ORA Security - The ORA Security document describes important aspects of security including identity, role, and entitlement management, authentication, authorization, and auditing (AAA), and transport, message, and data security.

ORA Application Infrastructure - Underpinning Oracle Fusion solutions and infrastructure is a computing platform that provides RASP qualities for enterprise-class computing. The ORA Application Infrastructure document describes these concepts and capabilities and defines the software platform on which solutions are built.

A.2 Other Resources

In addition, the following materials and sources of information relevent to SOA Infrastructure may be useful:

`http://www.oracle.com/products/middleware/index.html` - Oracle Fusion Middleware products homepage.

`http://download.oracle.com/docs/cd/E12839_01/index.htm` - Oracle Fusion Middleware Documentation.

`http://www.oracle.com/technology/products/ias/business_rules/index.html` - Oracle Business Rules Documentation.

`http://edocs.bea.com/wls/docs103/notes/new.html#wp1093291` - Oracle WebLogic Server Standards Support.

`http://download.oracle.com/docs/cd/E13157_01/wlevs/docs30/index.html` - Oracle CEP documentation.

`http://download.oracle.com/docs/cd/E10316_01/ouc.htm` - Oracle Univeral Content Management Documentation.

http://www.oracle.com/technology/products/integration/adapters/index.html - Oracle Integration Adapters documentation.

Glossary

4+1 Architectural View Model

4+1 is a view model designed by Philippe Kruchten for describing the architecture of software-intensive systems, based on the use of multiple, concurrent views. The views are used to describe the system in the viewpoint of different stakeholders, such as end-users, developers and project managers. The four views of the model are logical, development, process and physical view. In addition selected use cases or scenarios are utilized to illustrate the architecture serving as the 'plus one' view. Hence the model contains 4+1 views.

Business Process Execution Language (BPEL)

The Business Process Execution Language for Web Services (BPEL4WS or BPEL) is a formal, XML-based description language for business processes that interact via Web services.

Business Process Modeling Notation (BPMN)

BPMN is a graphical notation system for describing business processes in a business process diagram (BPD).

Consumer Contract

Usage Agreement or Consumer Contract refers to the Service contract between the consumer and the SOA (infrastructure or governing body) that specifies the terms of consumption of the Service. This is different from the Provider Contract, which governs the terms between the SOA and the Service provider.

J2EE Connector Architecture (JCA)

The J2EE Connector architecture provides a Java technology solution to the problem of connectivity between the many application servers and today's enterprise information systems (EIS).

Java Messaging Service (JMS)

The Java Message Service (JMS) API is an API for accessing enterprise messaging systems. It is part of the Java 2 Platform, Enterprise Edition (J2EE).

Java Portlet Specification (JPS)

JPS is a specification that defines a set of APIs to enable interoperability between portlets and portals, addressing the areas of aggregation, personalization, presentation, and security. JPS is based on JSR 168.

JSR 168

JSR-168 Portlet Specification standardizes how components for portal servers are to be developed.

JSR 268

JSR 268 is a next generation Java Portlet Specification that aligns the Java Portlet Specification with J2EE 1.4, other JSRs relevant for portlet programming, like JSR 188 and the next version of Web Services for Remote Portlets (WSRP).

Key Performance Indicator (KPI)

Key Performance Indicators (KPI) are measures or metrics used to help an organization define and evaluate how successful it is, in terms of making progress towards its short-term and long-term organizational goals.

Mediation

Mediation can be broadly defined as resolving the differences between two or more systems in order to integrate them seamlessly.

Message Exchange Pattern (MEP)

A Message Exchange Pattern (MEP) describes the pattern of messages required by a communications protocol to establish or use a communication channel.

Orchestration

Orchestration describes the automated arrangement, coordination, and management of Services.

Producer Contract

Producer Contract or simply the Service Contract refers to the Service contract between the provider and the SOA (infrastructure or governing body) that specifies the contract of the Service. The Producer Contract should be able to support the consolidated requirements of all the **Consumer Contract**s for any given Service.

Service Factory

Service Factory is an efficient organizational model to build enterprise class shared services using a centralized development orgnization. The Service Factory specializes in designing and building Services at rapid pace.

Source Code Management (SCM)

Source Code Management (SCM) is the management of changes to documents, programs, and other information artifacts.

Universal Description, Discovery and Integration (UDDI)

Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML based, open industry initiative, sponsored by the Organization for the Advancement of Structured Information Standards (OASIS), enabling businesses to publish service listings and discover each other and define how the services or software applications interact over the Internet.

Usage Agreement

Usage Agreement or **Consumer Contract** refers to the Service contract between the consumer and the SOA (infrastructure or governing body) that specifies the terms of consumption of the Service. This is different from the Provider Contract, which governs the terms between the SOA and the Service provider.

Web Service Remote Portlet (WSRP)

WSRP is a Web services standard that enables the plug-and-play of visual, user-facing Web services with portals or other intermediary Web applications. Being a standard, WSRP enables interoperability between a standards-enabled container and any WSRP portal.

XPDL

The XML Process Definition Language (XPDL) is a format standardized by the Workflow Management Coalition (WfMC) to interchange Business Process definitions between different workflow products, ie between different modeling tools and management suites. XPDL defines an XML schema for specifying the declarative part of workflow / business process.