

# **Oracle® Service Bus**

Concepts and Architecture

10g Release 3 (10.3)

October 2008

**ORACLE®**

Copyright © 2007, 2008, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

# Contents

## Introduction

Service-Oriented IT Trends . . . . .	1-1
Service-Oriented Architecture . . . . .	1-2
Service Bus Component of SOA . . . . .	1-5
Oracle Product Suite . . . . .	1-9
Oracle Service Bus . . . . .	1-10
Significance in an SOA Landscape . . . . .	1-12
Oracle Service Bus and the Service Lifecycle . . . . .	1-13
Functional Features and Benefits . . . . .	1-15

## Oracle Service Bus Architecture

Architecture Overview. . . . .	2-1
Key Architecture Concepts . . . . .	2-4
Message Processing . . . . .	2-4
Proxy Services . . . . .	2-6
Message-Flow Definitions . . . . .	2-7
Oracle Service Bus Core Features. . . . .	2-9
Service Integration . . . . .	2-9
Service Security . . . . .	2-13
Service Composition . . . . .	2-14
Service Management . . . . .	2-17
Oracle Service Bus Deployment . . . . .	2-18
Deployment Topology . . . . .	2-18

Distributed Configurations for Large-Scale Deployments . . . . .	2-19
Development, Staging, and Production Domains . . . . .	2-21
Configuration Metadata Export and Import. . . . .	2-21
Scripting Support . . . . .	2-23

## Service Integration

Message Brokering . . . . .	3-2
Services . . . . .	3-3
Service Types . . . . .	3-4
Transport Protocols . . . . .	3-5
Service Interfaces . . . . .	3-6
Messaging Models . . . . .	3-6
Message Formats . . . . .	3-7
Message Context . . . . .	3-7
Content Types. . . . .	3-9
Oracle Service Bus Resources. . . . .	3-9
Schemas and Data Types . . . . .	3-10
Transformation Maps . . . . .	3-11
JARs . . . . .	3-11
WSDLs . . . . .	3-11
Proxy Services . . . . .	3-12
Proxy Service Providers. . . . .	3-12
Alert Destinations . . . . .	3-12
JNDI Providers. . . . .	3-13
SMTP Servers . . . . .	3-13
Oracle Service Bus Security . . . . .	3-13
WS-Policies . . . . .	3-13
Service Accounts . . . . .	3-14

Security Levels . . . . .	3-15
Custom Security Credentials . . . . .	3-19

## Service Configuration

Resource Organization. . . . .	4-1
Project Explorer . . . . .	4-1
Projects and Folders . . . . .	4-2
Resource Cache . . . . .	4-3
Change Management . . . . .	4-3
Change Center . . . . .	4-3
Sessions . . . . .	4-4
Concurrent Modifications. . . . .	4-5
Tracking Configuration Changes . . . . .	4-6
Tracking Dependencies . . . . .	4-6
Semantic Integrity . . . . .	4-6
Reversing Changes to Resources and Sessions . . . . .	4-7
Service Discovery . . . . .	4-8
UDDI Registry . . . . .	4-8
Advantages of a UDDI Registry. . . . .	4-9

## Service Composition

Dynamic Content-Based Routing . . . . .	5-1
Business Services and Proxy Services . . . . .	5-2
Message Flow Modeling . . . . .	5-3
Message Pipelines . . . . .	5-4
Transformations. . . . .	5-10
Error Handling. . . . .	5-12
Message Validation. . . . .	5-12

# Service Management

Service Monitoring . . . . .	6-1
Dashboard . . . . .	6-1
Metric Aggregation . . . . .	6-2
SLA Enforcement via Alerts . . . . .	6-4
Message Reporting . . . . .	6-5

# Introduction

This section briefly discusses the current focus on service-driven IT strategies, key business drivers for Service-Oriented Architecture (SOA) initiatives and the strategic significance of an Enterprise Service Bus (ESB) component. It provides a conceptual overview of Oracle Service Bus infrastructure solution and functional capabilities that distinguish it as an SOA success factor. It is intended for management stakeholders responsible for SOA initiatives, integration-focused IT architects, service modelers or designers, and system administrators.

The following topics are included in this section:

- [“Service-Oriented IT Trends” on page 1-1](#)
- [“Oracle Product Suite” on page 1-9](#)
- [“Oracle Service Bus” on page 1-10](#)

## Service-Oriented IT Trends

In today’s highly competitive global market, businesses operate in a very liquid environment in which information is the most strategic asset. Responding rapidly to changes in competition, market dynamics, and regulatory mandates, with timely information, is critical for the effective functioning and overall success of businesses. To meet rapidly changing market demands, businesses have become increasingly service-driven, both in the ways they interact with customers and partners, and in how they design and build their IT infrastructure.

As businesses strive to deliver ROI through increased agility and responsiveness, they depend on enterprise IT groups to find new and cost effective means to deliver new services and to promote

the free flow of information and business processes within the organization. The following business drivers have made service-oriented IT architectures an economic reality in today's enterprise:

- Industry adoption of Web services to rapidly expose and enable new and legacy services
- Necessity to build system-centric processes spanning applications and users
- Necessity to quickly expose processes as services
- Execute mission-critical processes securely and consistently, with transactional integrity
- Develop fine-grained integrated application and process control
- Deliver high performance execution for straight-through processing

## Service-Oriented Architecture

Service-Oriented Architecture (SOA) has emerged as the leading IT agenda for infrastructure reformation, to optimize service delivery and ensure efficient business process management. Part of the paradigm shift of SOA are fundamental changes in the way IT infrastructure is designed—moving away from an application infrastructure to a converged service infrastructure.

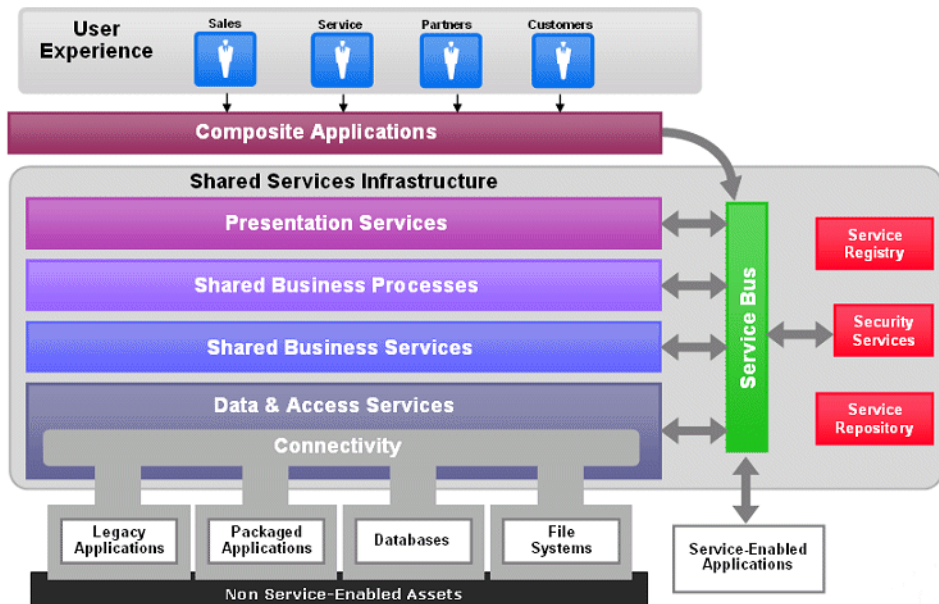
Service-Oriented Architecture enables discrete functions contained in enterprise applications to be organized as layers of interoperable, standards-based shared “services” that can be combined and reused in composite applications and processes.

In addition, this architectural approach also allows the incorporation of services offered by external service providers into the enterprise IT architecture. As a result, enterprises are able to unlock key business information in disparate silos, in a cost-effective manner. By organizing enterprise IT around services instead of around applications, SOA helps companies achieve faster time-to-service and respond more flexibly to fast-paced changes in business requirements.

In recent years, many enterprises have evolved from exploring pilot projects using ad-hoc adoption of SOA and expanded to a defined repeatable approach for optimized enterprise-wide SOA deployments. All layers of an IT SOA architecture have become service-enabled and comprise of presentation services, business processes, business services, data services, and shared services.



Figure 1-1 SOA Conceptual Architecture



## Service Mediation Challenges

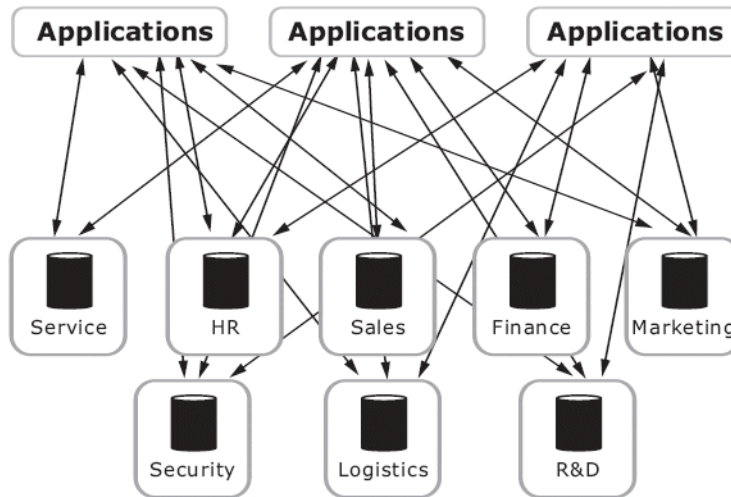
A major challenge for SOA initiatives is attributed to the inherently heterogeneous multi-vendor IT landscape in many enterprises, and the resultant individual silos of business information. Rather than incur the cost and complexity of replacing disparate components of legacy infrastructure, enterprises often choose to extend existing business applications as services for use in other business processes and applications.

The influx of Web service interfaces to functionality within existing packaged applications, often introduces services that do not adhere to established service and compliance guidelines. This is especially true if the services are published from core enterprise systems such as CRMs, Data Warehouses, and ERPs.

In the absence of robust and comprehensive service infrastructure solutions, developers have used a variety of “middleware” technologies to support program-to-program communication, such as object request brokers (ORBs), message-oriented middleware (MOM), remote procedure calls (RPC). More recently, IT infrastructure developers hard-coded complex integration logic as point-to-point connections to web services, in order to integrate disparate applications and

processes. This inevitably resulted in complex service sprawls within enterprise IT environments. The following figure illustrates a typical static service integration scenario.

**Figure 1-2 Service Sprawl Challenge**



The following are other service related challenges attributed to heterogeneous IT architectures:

- Tightly-coupled business services integration due to complex and rigid hard-wired connections
- Difficulty managing deployed services due to disparate protocols and applications involved
- High total cost of ownership for the enterprise
- Impaired ability to reuse services
- Inherent replication of transport, transformation, security, and routing details
- Exponential redevelopment and redeployment efforts when service end-point interfaces change
- Inevitable service disruption that significantly impact service consumers

Enterprise architects and web service modelers with goals to streamline IT infrastructure now require enterprise service capabilities that address the following IT needs:

- Simplify access and updates to data residing in different sources

- Reuse services developed across the enterprise and effectively manage their lifecycle
- Provide dynamic configuration of complex integration logic and message routing behavior
- Enable run-time configuration capabilities into the service infrastructure
- Ensure consistent use of the enterprise services
- Ensure enterprise services are secure and comply with IT policies
- Monitor and audit service usage and manage system outages

## Composite Applications and Service Layering

In an SOA initiative, composition is an integral part of achieving business flexibility through the ability to leverage existing assets in higher-order functions. Within a mature SOA environment, complete business applications are composed using existing services to quickly meet the business needs. Flexibility in the service provisioning process, is achieved by avoiding coding logic in service implementations.

Many organizations develop services at very granular levels and the proliferation of many small specific services are difficult to compose into broader logical services. Layering of Services is as a way of breaking out of the limitations of monolithic applications and shortening development, release and test cycles. By defining a layered approach to service definition and construction, the service infrastructure team can achieve the right mix of granular and course-grained services required to meet their current and future business demands.

Service Layers typically comprise of the following services:

- Physical Services: that may represent functions that retrieve data in its raw form
- Canonical Services: that may define a standard view of information for the organization, leveraging industry-standard formats and supporting a very wide data footprint
- Logical Services: that provide a more client-specific granular view of information, generated at compile time using highly-optimized queries
- Application Services: that are consumed directly by applications in a line-of-business dependent fashion and may be exposed through presentation services

## Service Bus Component of SOA

The core of SOA success depends on an Enterprise Service Bus (ESB) that supports dynamic synergy and alignment of business process interactions, continual evolution of existing services

and rapid addition of new ones. To realize the benefits of SOA, it is imperative that IT organizations include a robust and intelligent service intermediary that provides a layer of abstraction to mask the complexities of service integration in heterogeneous IT environments, typical in today's enterprises. While an intermediary layer of abstraction previously implied a platform for customizing enterprise applications, today it implies toolkits for service customization and scalable infrastructures that support loosely coupled service interactions with a focus on service mediation.

**Figure 1-3 Enterprise Service Bus**



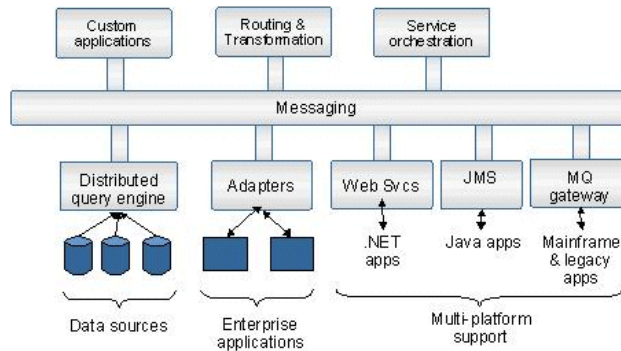
ESBs have been instrumental in the evolution of integrated middleware infrastructure technology by combining features from previous technologies with new services, such as message validation, transformation, content-based routing, security and load balancing. ESBs use industry standards for most of the services they provide, thus facilitating cross-platform interoperability and becoming the logical choice for companies looking to implement SOA.

An ESB provides an efficient way to build and deploy enterprise SOA. ESB is a concept that has gained the attention of architects and developers, as it provides an effective approach to solving common SOA hurdles associated with service orchestration, application data synchronization, and business activity monitoring. In its most basic form, an ESB offers the following key features:

- Web services: support for SOAP, WSDL and UDDI, as well as emerging standards such as WS-Reliable Messaging and WS-Security
- Messaging: asynchronous store-and-forward delivery with multiple qualities of service
- Data transformation: XML to XML
- Content-based routing: publish and subscribe routing across multiple types of sources and destinations

- Platform-neutral: connect to any technology in the enterprise, e.g. Java, .Net, mainframes, and databases

**Figure 1-4 ESB Architecture**



A robust SOA suite offers:

- Adapters, to enable connectivity into packaged and custom enterprise applications, as well as leading technologies.
- Distributed query engine, for easily enabling the creation of data services out of heterogeneous data sources
- Service orchestration engine, for both long-running (stateful) and short-running (stateless) processes
- Application development tools, to enable the rapid creation of user-facing applications
- Presentation services, to enable the creation of personalized portals that aggregate services from multiple sources

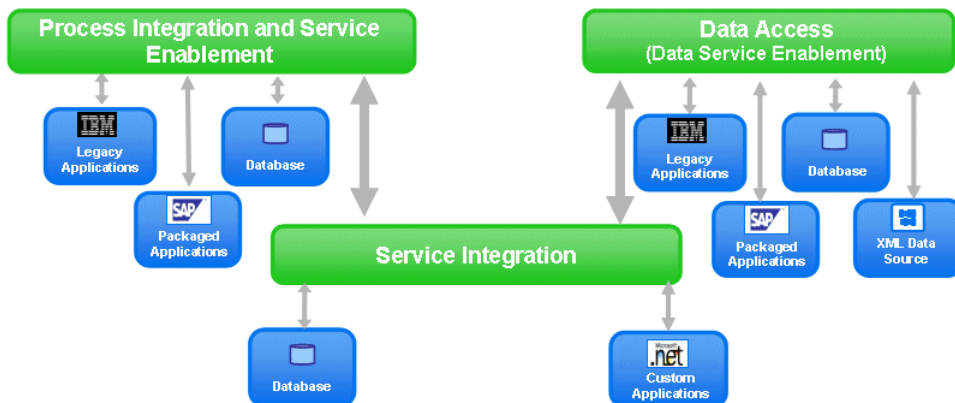
Using ESBs offers greater flexibility for enterprises to connect heterogeneous resources, by eliminating the need for brittle high-maintenance point-to-point connections. Adding an ESB intermediary between service consumers and service providers, shields them from the implementation details of underlying service end-point interfaces, reducing or eliminating the redevelopment and redeployment impacts at the service-consumer level.

Best in class enterprises have achieved SOA success by harnessing high-speed enterprise-ready ESB intermediaries that strategically integrate service mediation capabilities and business process management functionality. Recognizing the significance of operational service management as a critical SOA success factor, they have implemented solutions that provide

enterprise-class service scalability, reliability, customization and security. By adopting such solutions built specifically for management and governance of an SOA service lifecycle, these enterprises have obtained the following business benefits:

- Minimized costs by accelerating SOA deployment initiatives
- Ensured customer satisfaction by assurance of continuous service availability
- Insulated service consumers to changes in service infrastructure by virtualizing service end points
- Maximized ROI by leveraging shared services infrastructure and using consistent modeling methodologies
- Reduced integration burden by simplifying service interactions
- Improved effectiveness of SOA initiatives through accurate run-time governance of shared services
- Justification of SOA spending by inventory and tracking of run-time services
- Accurate cost benefit decisions by measuring the benefit or cost avoidance obtained through SOA

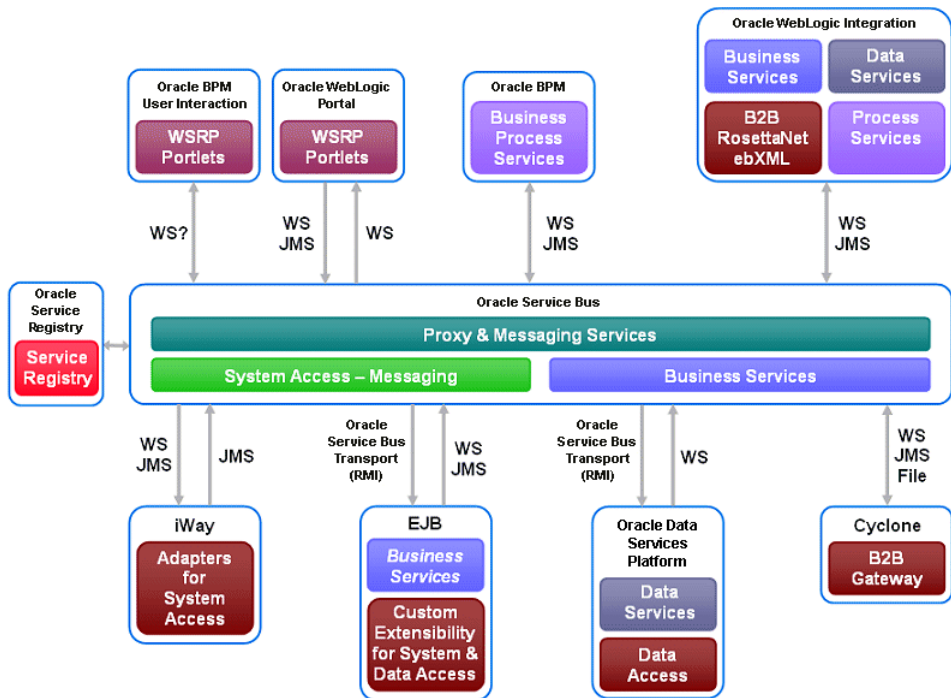
**Figure 1-5 Enterprise Integration for SOA**



# Oracle Product Suite

Oracle offers the first service-infrastructure product family built from the ground up for Service-Oriented Architectures, giving IT a unified set of products to successfully deploy an SOA across their organization and achieve better business agility and IT efficiency.

**Figure 1-6 Oracle Service Bus Shared Services Product Architecture**



The following product lines are available within the Oracle product family:

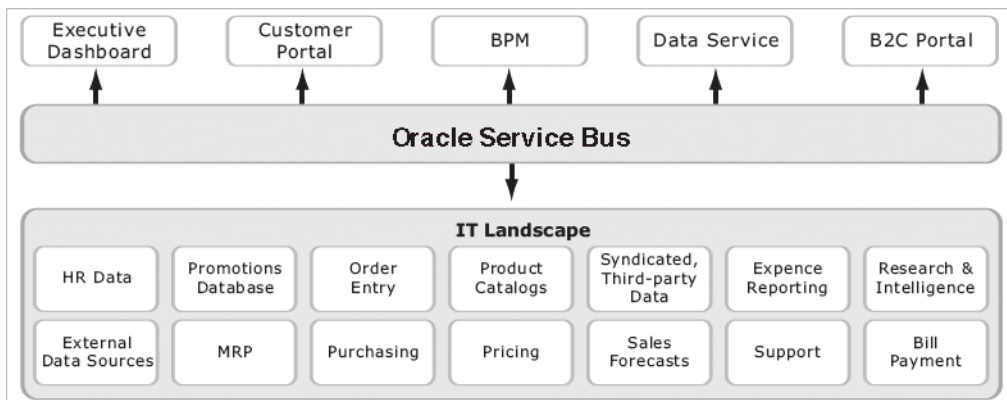
- **Oracle User Interaction:** an integrated product line used to create a variety of interactive solutions that leverage the service infrastructure, including portals and composite applications.
- **Oracle Business Process Management:** a product line that includes software that automates, executes, and monitors the entire lifecycle of a business process.
- **Oracle Data Service Integrator:** a product line which enables data services that deliver unified, real-time views of data from disparate sources across the enterprise.

## Oracle Service Bus

Oracle Service Bus is a proven market-leading Enterprise Service Bus (ESB) built from the ground up for SOA lifecycle management that provides foundation capabilities for service discovery and intermediation, rapid service provisioning and deployment, and governance.

This service-infrastructure software adheres to the SOA principles of building coarse grained, loosely coupled, and standards-based services, creating a “neutral container” in which business functions may connect service consumers and back-end business services, regardless of underlying infrastructure. The following figure illustrates the role of Oracle Service Bus as a service intermediary in an enterprise IT SOA landscape:

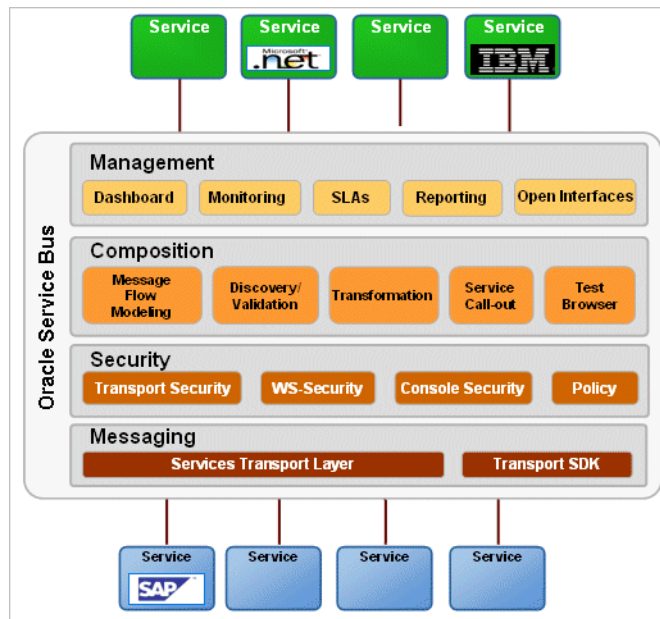
**Figure 1-7 Oracle Service Bus Intermediary**



Built to meet exacting standards for reliability, availability, scalability, and performance, Oracle Service Bus uniquely combines the integration capabilities of an Enterprise Service Bus with operational service management, into a single enterprise-class software product, with a layered functional architecture.



Figure 1-8 Oracle Service Bus Functional Features



The functional features of Oracle Service Bus can be categorized into the following functional layers:

- **Messaging Layer:** that reliably connects any service by leveraging standards web service transports, traditional messaging protocols and configuration of enterprise-specific custom transports. For more information, see [“Adaptive Service Messaging Layer”](#).
- **Security Layer:** a rapid service configuration and integration environment that abstracts policies associated with routing rules, security, and service end-point access. For more information, see [“Optimized Pluggable Security Layer”](#).
- **Composition Layer:** a meta-data driven feature-rich configuration interface for service discovery and validation capabilities for automatic import and synchronization of services with UDDI registries, allows message flow modeling, transformations, third-party service callouts and a test console. For more information, see [“Rich Service Composition Layer”](#).
- **Management Layer:** a service management environment that includes dynamic service and policy configuration, centralized usage and performance monitoring, and management of services - not just Web services, but also Java, .Net, messaging services, and legacy end points. For more information, see [“Embedded Service Management Layer”](#).

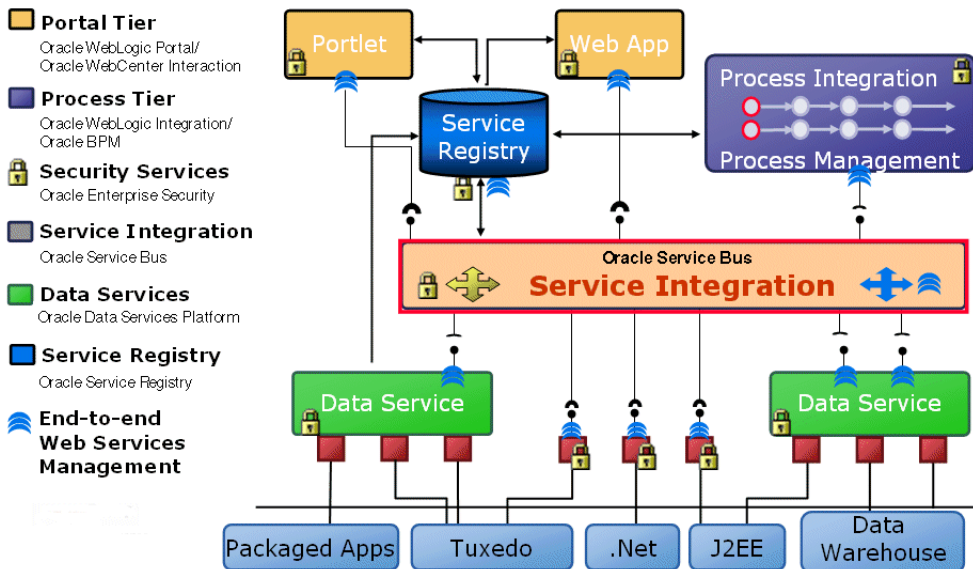
## Significance in an SOA Landscape

Oracle Service Bus is at the heart of Oracle's comprehensive business integration solution and belongs to the Oracle Messaging product line. Oracle Service Bus is primarily targeted for managing different types of services, and providing traditional message brokering across heterogeneous IT environments.

The lightweight, stateless, high-performance architecture of Oracle Service Bus and its converged intelligent message mediation and service lifecycle management capabilities, make it an ideal core component of distributed services networks.

It is designed to fit into the broader IT Service-Oriented Architecture (SOA) landscape as a distributed service management intermediary and can be integrated with other Oracle business process management solutions in distributed heterogeneous deployments.

**Figure 1-9 Oracle Service Bus Significance in SOA Architecture**



## Oracle Service Bus Use Cases

Oracle Service Bus is a powerful lightweight, cost-effective technology that can be used by service developers and architects for the following use cases:

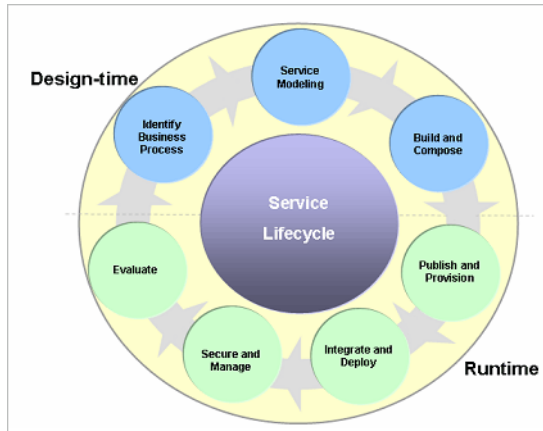
- **Building an Enterprise-Wide SOA:** a backbone that transports and routes messages across an enterprise, for configuring request-and-response message flows between service consumers and service providers
- **Building reusable atomic services:** that facilitate organizational flexibility and promote application integration and data synchronization across multiple applications
- **Creating composite applications:** by rapidly building new applications that access services of existing applications from a shared service catalog, reducing time to market through reuse
- **Business Activity Monitoring (BAM):** enable business users to get access to key performance indicators and act on business alerts, listen for business events flowing through the infrastructure, and orchestrate services in response to these events, using portals that provide a personalized view of enterprise services

## Oracle Service Bus and the Service Lifecycle

The lifecycle of a service comprises of the following two phases:

- **Design phase:** in which the service architecture team identifies an organization's business needs and models a number of services and application interfaces to support those needs
- **Run-time phase:** in which the services modeled using the catalog of business needs are used as a roadmap for service creation and exposed as run-time offerings within the organization

**Figure 1-10 Services LifeCycle**



Oracle Service Bus plays an integral part in the service lifecycle run-time phase. It facilitates the following important functions in the services lifecycle:

- Promotes logical or conceptual layering of the system - by allowing for design and provisioning layered services during the run-time phase. This ensures run-time flexibility and no loss of agility at provisioning time.
- Manages and monitors the flow of messages between consumers and providers
- Insulate users and processes from service changes - by abstracting services and removing shared integration logic from service endpoints
- Provides service transformation, validation, enrichment and routing - by bridging protocols, message styles, security and data formats
- Provides visibility and operational service management - by exposing services for use by consumers

## **Role of Oracle Service Bus in a Service Cycle**

A key design philosophy of Oracle Service Bus is the physical separation of management functions from service implementations. As part of an enterprise's messaging fabric, Oracle Service Bus can be used horizontally integrate many applications and systems, spanning service implementations in different departments built by different teams.

As services are created, they are registered and exposed for later consumption by other services or processes. Services can be registered directly with Oracle Service Bus in its local service

registry, or imported from an enterprise service registry such as the Oracle Service Registry. After services are registered with Oracle Service Bus, it configures proxy interfaces that define the message flow for communicating with these services.

This flow contains any transformation and security requirements that must be applied, as well as specifications for routing the message to the service. After services are registered with Oracle Service Bus, business processes, such as those created with Oracle WebLogic Integration, can consume these services and orchestrate them to support various business contexts. These orchestrated processes define how the services are used and applied to business requirements and fine-grained business processes. These business processes are then exposed for use by end users through a user interface (UI), which could be a transactional portal such as Oracle WebLogic Portal or a collaborative portal such as Oracle User Interaction.

Oracle Service Bus again steps into the lifecycle to monitor and manage message flow, system health, and availability between service end points. This information may be reported to business and operations analysts who can analyze it for patterns of behavior indicating where improvements should be made. The lifecycle begins again as services evolve over time and new versions are released.

## Functional Features and Benefits

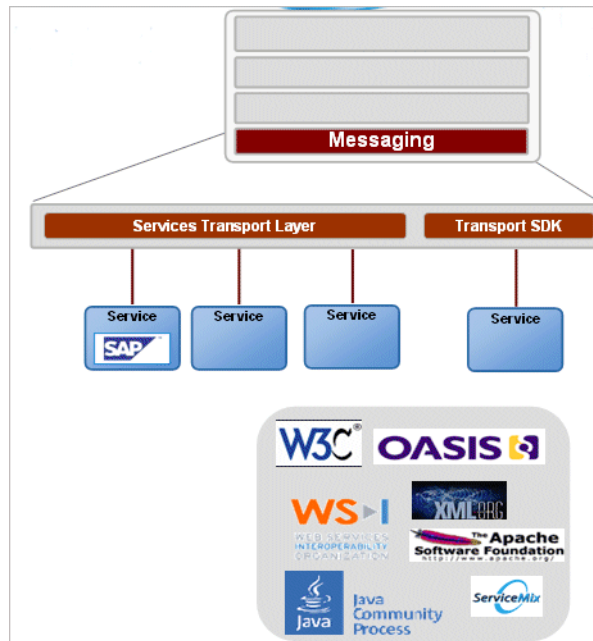
The key functional capabilities in the Oracle Service Bus can be classified into the following functional layers:

- “Adaptive Service Messaging Layer”
- “Optimized Pluggable Security Layer”
- “Rich Service Composition Layer”
- “Embedded Service Management Layer”

### Adaptive Service Messaging Layer

Oracle Service Bus supports an unprecedented level of heterogeneity and can reliably connect any service by leveraging standards. Existing middleware, applications, and data sources become first-class citizens of SOA initiatives, protecting existing IT investments and enabling IT to connect, mediate, and manage services using heterogeneous end points, formats, and protocols.

**Figure 1-11 Adaptive Service Messaging Layer**



Oracle Service Bus works with diverse Web Services transports including the following:

- HTTP/SOAP
- WS-I, WS-Security, WS-Policy, WS-Addressing
- SOAP v1.1 and v1.2
- EJB/RMI on WebSphere

Oracle Service Bus includes support accepting SOAP 1.1 messages and converting them to SOAP 1.2 messages when necessary to invoke a SOAP 1.2 business service, and vice versa.

Oracle Service Bus promotes efficient message orchestration by working with traditional messaging protocols and messaging paradigms. For a complete list of supported protocols and communication paradigms, see [Chapter 3, “Service Integration”](#), topics [“Transport Protocols”](#) and [“Messaging Models”](#).

In addition to its industry-leading support for Web services, Oracle Service Bus also provides native connectivity to MQ Series, CICS, .NET, C/C++ applications. It allows creation and configuration of enterprise-specific custom transports using the Custom Transport Software Development Kit (SDK) and native transport for Oracle Data Service Integrator. It provides the ability to create a generic proxy services that can accept any SOAP or XML message, using generic proxy service templates.

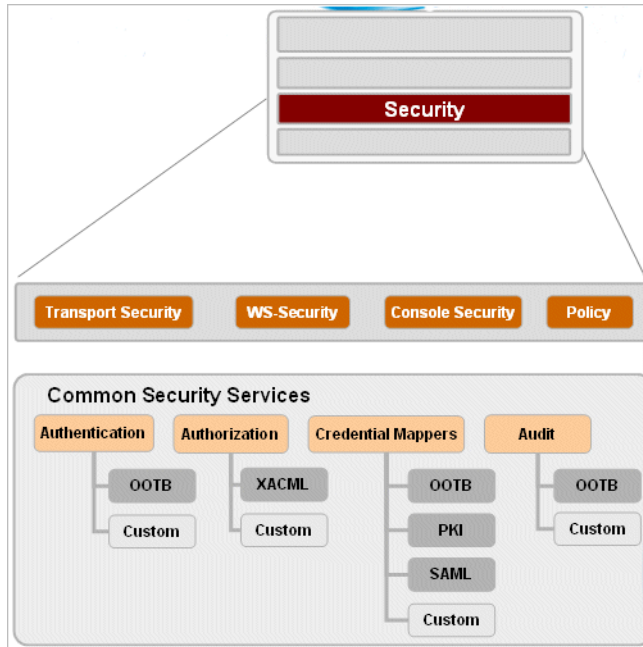
Oracle Service Bus supports optimized database queries across the SOA for high performance and reliability, and interoperability with web service integration technologies including .NET, Tibco EMS, IBM MQ, IBM WebSphere, Apache Axis, Cyclone B2B Interchange, and iWay adapters.

For information on Oracle Service Bus interoperability, see [Oracle Service Bus Product Support Information](#) and [Oracle Service Bus Interoperability and Transports](#) documentation.

## Optimized Pluggable Security Layer

Oracle Service Bus ensures service security at all levels. A comprehensive set of components for built-in security give customers significant flexibility and choice. Users can also plug in home-grown or third-party security components. Built-in capabilities allow flexibility in implementation by enabling security at the following levels:

- Transport Security - SSL/Basic Auth
- Message Security - support for WS-Policy/WS-Security, SAML, UserID/Password, X509, Signing & Encryption, and Custom security credentials
- Console Security - support for user Web Single-Sign-On and role based access
- Policy - leverages WS-Security and WS-Policy

**Figure 1-12 Optimized Pluggable Security Layer**

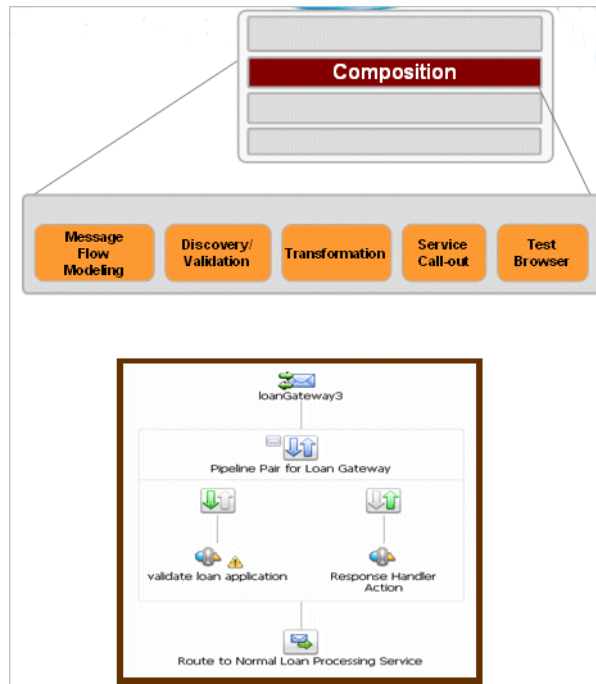
For more information on Oracle Service Bus security features, see [Oracle Service Bus Security Guide](#).

For more information on Oracle Service Bus supported standards, see [Supported Standards and Security Providers](#) in *Oracle Service Bus Security Guide*.

## Rich Service Composition Layer

Oracle Service Bus provides a rich environment for service configuration, modeling, discovery and message validation. It also supports message transformations, service callouts to external service providers, and a test browser.



**Figure 1-13 Service Composition Layer**

Oracle Service Bus configuration-driven composition environment, with Workshop for WebLogic and the Oracle Service Bus Console, is designed with a no coding approach, allows message flow modeling using versatile graphical modeling tools. The modeling capabilities allow for configuration of dynamic content-based routing, message validation and exception handling. For more information on dynamic routing, see [Chapter 5, “Service Composition”](#), topic [“Dynamic Content-Based Routing”](#).

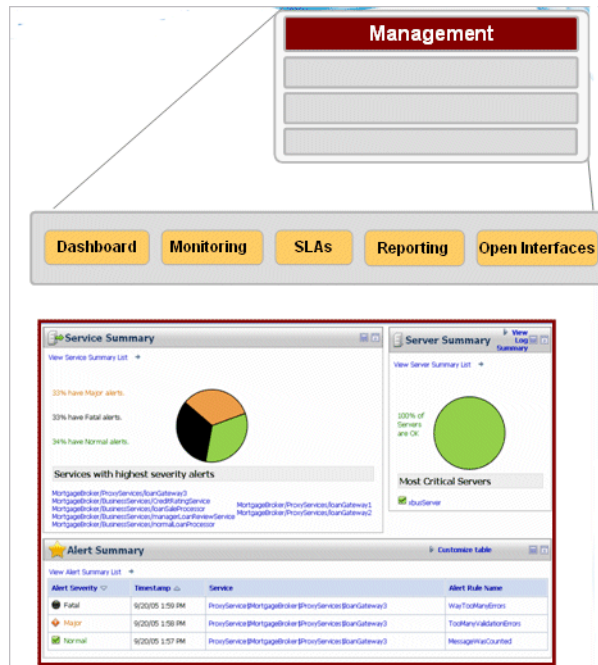
The composition environment includes a service discovery and validation capabilities for automatic import and synchronization of services with UDDI registries. This functionality allows for validation to ensure service integrity and reconciliation of conflicts before deployment. For more information on UDDI registries, see [Chapter 4, “Service Configuration”](#), topic [“UDDI Registry.”](#)

Oracle Service Bus supports XML and non-XML transformation of disparate data types shared between source and destination services. Data mappings using XQuery and the eXtensible Stylesheet Language Transformation (XSLT) standard are supported. These can be created externally and imported into Oracle Service Bus, or scripted using XQuery in the Oracle Service Bus Console. Message content reformatting can be performed using XQuery or XSLT or by manipulating message content by adding, removing, or replacing selected elements. For more information on transformations, see [Chapter 5, “Service Composition”](#), topic [“Transformations.”](#).

Oracle Service Bus supports dynamic routing through a service callout feature which allows more flexible and sophisticated message flow modeling. Service callouts within message flows can be used to invoke other services registered within Oracle Service Bus and may be Java exit (Plain Old Java Object) or Web Services call-outs. The composition environment also features a test console for tracing and trouble-shooting services. For more information on service callouts, see [Constructing Service Callouts](#) in *Oracle Service Bus User Guide*.

## Embedded Service Management Layer

Oracle Service Bus offers embedded service management capabilities that provide optimized governance of all messaging. Its preemptive support ensures that mission-critical business processes continue to serve customer needs, even as business demands, requirements, and workloads change.

**Figure 1-14 Oracle Service Bus Embedded Service Management**

Oracle Service Bus Dashboard provides a unified data service interface for all application development and maintenance, service monitoring and management and improved operations support. The dashboard allows for monitoring of fault and performance metrics, viewing of summaries for aggregated ESB. It allows for dynamically defining and managing routing relationships, transformations, and policies. For more information on Dashboards, see [Chapter 6, “Service Management”](#), topic “Dashboard”.

Oracle Service Bus has built-in optimizations for performance and monitoring, guarantee quality of service through alert monitoring on single node or entire server, and configurable Service Level Agreements (SLAs) for messages. It also supports viewing and managing SLA alerts on operation metrics and message pipelines. For more information on configuring SLAs, see [Chapter 6, “Service Management”](#), topic “SLA Enforcement via Alerts”.

Oracle Service Bus allows integration of widely adopted out-of-the-box third-party reporting tools as well as custom Enterprise Systems Management frameworks. In addition, it supports open interfaces for operational and deployment customization, JMX monitoring interfaces and SNMP Alerts. For more information on reporting features, see [Chapter 6, “Service Management”](#), topic “Message Reporting”.

## Feature Benefits

The following table summarizes functional features of the Oracle Service Bus and highlights the business requirements addressed by each functionality.

**Table 1-1 Oracle Service Bus Features and Benefits**

<b>Functionality</b>	<b>Functional Feature</b>	<b>Business Benefit</b>
Message Routing	Configuration-driven intelligent, content-based and identity-based routing	Rapidly respond to business needs by quickly configuring routing rules based on changes to business rules or existing IT systems, without coding
Message Transformation	Dynamic message transformation based on XQuery or XSLT, supporting multiple message formats	Flexibly adapt to evolving SOA and integration project scenarios through the ability to dynamically transform and route services using simple and/or complex routing rules and/or message payloads
Service Registry	Automated or administrator-driven interoperability with UDDI V3 registries for service publishing and reuse	Increase ease of re-use by automatically discovering existing services and exporting new services to the service registry
Service Provisioning	Simplified service provisioning	Increase ease of managing multiple versions of services, simplify and speed deployments by eliminating build-test development cycles
Message Security	Optimized, pluggable, policy-driven transport and message level security	Leverage existing investments in security infrastructure and seamlessly broker between multiple security frameworks
Service End-point Interoperability	Extensibility and expanded service end-point support	Extend solution to accommodate unique IT requirements using infrastructure with certified interoperability with multiple standards, protocols, and vendors
Service Level Agreements	Rules-driven, configurable Service Level Agreement (SLA) enforcement	Gain visibility and control by enabling users to set SLAs based on a number of factors and alerts when the SLAs are not met

Functionality	Functional Feature	Business Benefit
Message Transport	Extensible support for heterogeneous transports between service end points including custom transports via the Custom Transport SDK	Provides flexibility to leverage existing investments in disparate systems and/or ensure smooth transition from older to newer systems
Message Brokering	WS-I compliant Intelligent messaging brokering with support for multiple transport types, message formats	Ensure investment protection and leverage existing infrastructure through the ability to orchestrate services from existing IT systems with disparate messaging protocols without needing to change the systems and styles
Service Availability	Proactive infrastructure health and availability monitoring with JMX and SNMP	Maintain health and availability of the SOA through easy configuration of support of performance metrics and SLAs using a built-in, feature-rich dashboard OR 3rd party performance management systems.
Service Monitoring Dashboard	Flexible, graphical, and embedded management and monitoring dashboard	Automatically monitor and manage status of performance metrics and SLAs using a built-in, feature-rich dashboard or 3rd party performance management systems. Proactively take corrective action based on alerts.
Service Deployment	Easy, customizable programmatic or console-driven deployment	Ability to enforce governance and speed deployments

## Related Topics

- [Supported Standards and Security Providers in \*Oracle Service Bus Security Guide\*.](#)
- [Using the Oracle Service Bus Console](#)
- [Supported Configurations for Oracle Service Bus](#)
- [Oracle Service Bus User Guide](#)
- [Oracle Service Bus Interoperability Support Matrix](#)
- [Oracle Service Bus Interoperability and Transports](#)

## Introduction

# Oracle Service Bus Architecture

This section provides an architectural overview of Oracle Service Bus and highlights operational features that enable rapid service integration, provisioning, and management across a heterogeneous IT infrastructure. It is intended for integration-focused IT architects responsible for messaging and service oriented architectures (SOA). It includes the following sections:

- [“Architecture Overview” on page 2-1](#)
- [“Key Architecture Concepts” on page 2-4](#)
- [“Oracle Service Bus Core Features” on page 2-9](#)
- [“Oracle Service Bus Deployment” on page 2-18](#)

## Architecture Overview

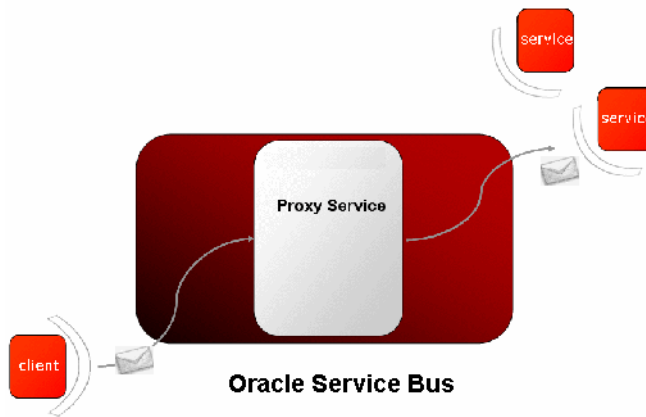
Oracle Service Bus architecture is centered around an Enterprise Service Bus. The bus provides message delivery services, based on standards including SOAP, HTTP and Java Messaging Service (JMS). It is typically designed for high-throughput, guaranteed message delivery to a variety of service producers and consumers. It supports XML as a native data type, while also offering alternatives for handling other data types.

Oracle Service Bus is policy driven and enables you to establish loose coupling between *service clients* and *business services*, while maintaining a centralized point of security control and monitoring. It stores persistent policy, proxy service, and related resource configurations in metadata, that can be customized and propagated from development through staging to

production environments required. The message-brokering engine accesses this configuration information from its metadata cache.

Oracle Service Bus is an intermediary that processes incoming service request messages, determines routing logic, and transforms these messages for compatibility with other service consumers. It receives messages through a transport protocol such as HTTP(S), JMS, File, and FTP, and sends messages through the same or a different transport protocol. Service response messages follow the inverse path. The message processing by Oracle Service Bus is driven by metadata, specified in the *message flow definition* of a proxy service.

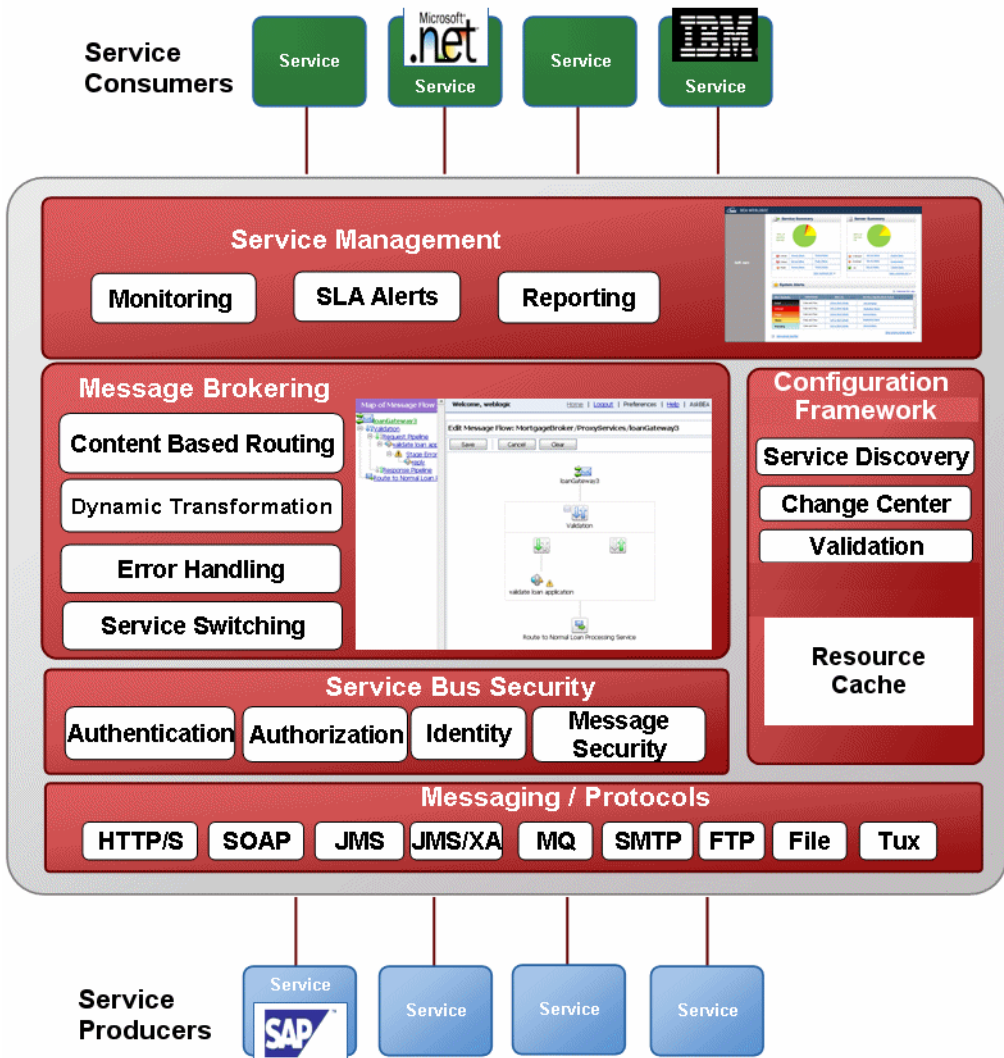
**Figure 2-1 Oracle Service Bus Service Interactions**



The following high-level architecture diagram illustrates Oracle Service Bus and its functional subsystems: service management, message brokering, configuration framework, security and transport layer, and messaging protocols.



Figure 2-2 Oracle Service Bus High-Level Architecture

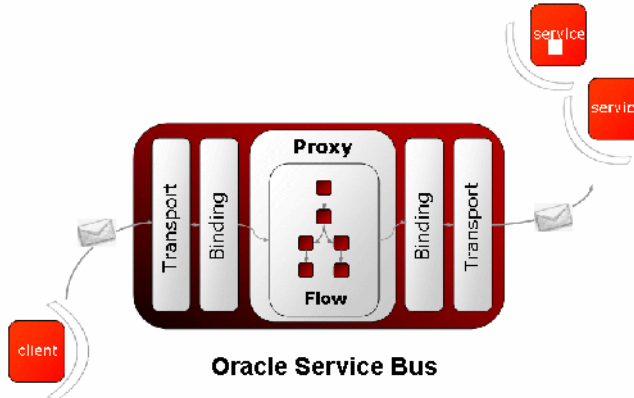


## Key Architecture Concepts

### Message Processing

Messages can contain data or status information about application processes, or instructions for the recipient, or both. Oracle Service Bus enables you to route messages based on their contents and to perform transformations on that content. The processing happens through the transport and binding layers of Oracle Service Bus.

**Figure 2-3 Binding and Transport Layers in Oracle Service Bus**

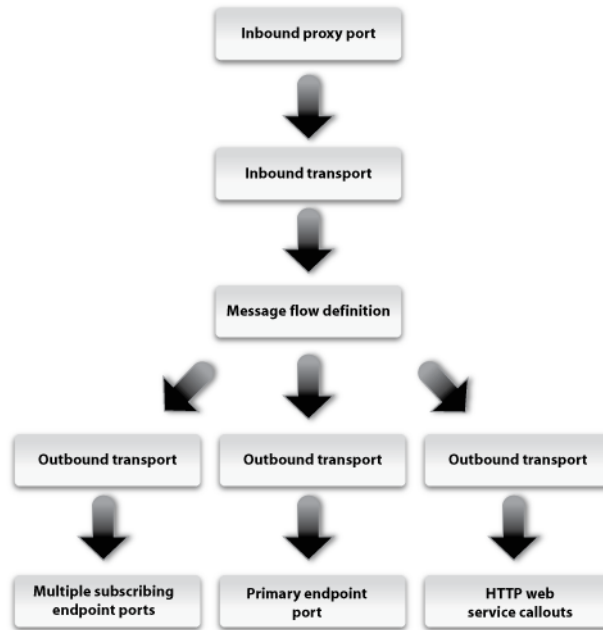


The processing of messages occurs in the following sequence of events:

1. Processing of the inbound transport
2. Message flow execution
3. Processing of the outbound transport

After a message is sent to an endpoint (either a business service or another proxy service), Oracle Service Bus processes the response message in a similar model as that described in the preceding sequence of events.

The following figure illustrates a high-level message flow process through the Oracle Service Bus, from inbound endpoint (proxy service) to outbound endpoint (service transport URL - a business service or another proxy service).

**Figure 2-4 Oracle Service Bus Message Processing**

The following sections describe each layer involved in this message processing.

## Binding Layer

The binding layer:

- packs and unpacks messages as necessary
- handles security for messages
- hands messages off to start the message flows (request and response)

## Transport Layer (Inbound)

The inbound transport layer is the communication layer between client services (or service consumers) and Oracle Service Bus. It is responsible for handling communication with the service client endpoint and acts as the entry point for messages into Oracle Service Bus. The inbound transport layer primarily deals with raw bytes of message data in the form of input/output streams.

It provides support for compatible transport protocols, including HTTP(S), JMS, FTP, File, and E-mail. It is not involved in data processing but is responsible for returning response messages to service consumers and handles meta-data for messages, including endpoint URIs, transport headers, etc.

## Transport Layer (Outbound)

The outbound transport layer is responsible for the communication between business services (or service producers) and Oracle Service Bus. It is responsible for moving messages from Oracle Service Bus to the business service or proxy service and for receiving the response from the services. The message data, at the transport level, is in raw bytes in the form of input/output streams. The outbound transport layer provides support for compatible transport protocols, including HTTP(S), JMS, FTP, File, and E-mail. It is not involved in data processing but handles meta-data for messages, including endpoint URIs, transport headers, etc.

## Proxy Services

Proxy services are a fundamental concept in the architecture of Oracle Service Bus. They are the interface that service consumers use to connect with managed back-end services. Proxy services are definitions of intermediary Web services that the Service Bus implements locally. Oracle Service Bus Console allows configuration of a proxy service by defining its interface in terms of Web Services Description Languages (WSDLs) and the type of transport it uses. Message processing logic is specified in *message flow definitions* when defining a proxy service. For more information on proxy services, see [“Proxy Services” on page 5-2](#).

## Message Context

The context of a proxy service is a set of XML variables that are shared across the request flow and response flow. New variables can be dynamically added or deleted to the context. Predefined context variables contain information about the message, transport headers, security principles, metadata for the current proxy service, and metadata for the primary routing and publishing services invoked by the proxy service.

The context can be read and modified by XQuery expressions and updated by transformation and in-place update actions. The core of the context contains the variables \$header, \$body, and \$attachments. These wrapper variables contain the Simple Object Access Protocol (SOAP) header elements, SOAP body element, and Multipurpose Internet Mail Extensions (MIME)

attachments, respectively. The context gives the impression that all messages are SOAP messages, and non-SOAP messages are mapped to this paradigm.

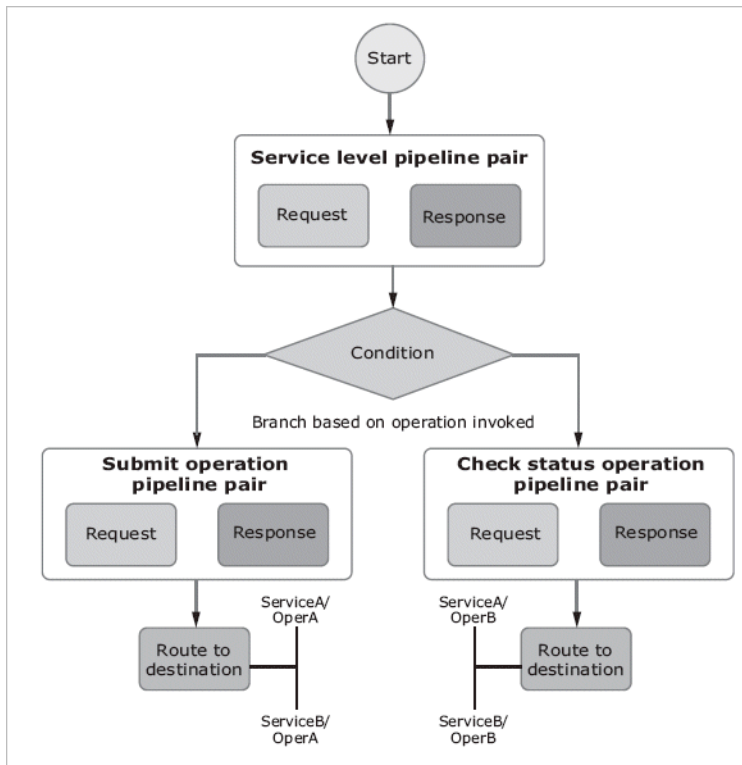
Since a proxy service can route messages to multiple business services, a proxy service can be configured with an interface that is independent of the business services it communicates with. Using generic proxy templates, the proxy service can be configured as a message-flow definition that dynamically routes messages to appropriate business services based on content-based routing logic. A proxy service can also map message data into appropriate protocol formats required by the end-point business service, allowing for dynamic run-time protocol switching.

For more information on proxy templates, see [Service Composition](#) in the *Concepts and Architecture Guide*.

## Message-Flow Definitions

The implementation of a proxy service is specified by a *message flow definition*. The message flow defines the flow of request and response messages through the proxy service. The following four elements are used to construct a message flow:

- A *pipeline pair*, one for the request and one for the response. The pipelines consist of a sequence of stages that specify actions to perform during request or response processing.
- A *branch node* to branch based on the values in designated parts of the message or message context or to branch based on the operation invoked.
- A *route node* used to define the message destination. The default route node is an *echo node* that reflects the request as the response.
- A *start node*.

**Figure 2-5 Sample Message Flow**

Message flow elements can be combined in arbitrary ways to form a tree structure with the *start node* always (and only) occurring as the root of the tree and the route nodes. The last nodes in a branch (leaf nodes) may be route nodes or echo nodes.

The request message starts at the start node and follows a path to a leaf node, executing actions in the request pipelines. If the leaf is a route node, a response is generated (could be empty if the service is a one way service). If the leaf is an echo node, the request is also considered to be the response. The response follows the inverse path in the tree skipping actions in the branch nodes, but executing actions in response pipelines.

A response is then finally sent from the top of the tree, if the interface or operation was request/response; otherwise the response is discarded. For more information on message flow definitions, see [Modeling Message Flow in Oracle Service Bus](#) in *Oracle Service Bus User Guide*.

A set of transformations that affects context variables can be defined before the message is sent to the selected endpoint or after the response is received. A Web services callout can be an alternative to an XQuery or XSLT transformation to set the context variable. For information on how to configure Oracle Service Bus transformation maps, see [Transforming Data Using the XQuery Mapper](#).

WS-Security processing as well as authorization is transparently performed at the Start node, when invoking a business service with a WS-policy. For information on how to configure Oracle Service Bus security, see the [Oracle Service Bus Security Guide](#).

## Oracle Service Bus Core Features

By fusing the concepts of the ESB, message brokering, and operational services management into a single product, Oracle Service Bus allows management and integration of messages and services across a services network. Its core functional features are separated into the following categories:

- **Service Integration** - features used for integrating disparate service end-points, message brokering, and mediating and exposing services for reuse
- **Service Security** - features used for service authentication and authorization, message security enforcement, and user identity validation
- **Service Composition** - features used for configuring message routing logic, message transformation, service configuration, validation and registry
- **Service Management** - features used for monitoring and managing service activity and availability

## Service Integration

### Communication Types

To support heterogeneous environments, Oracle Service Bus accommodates multiple messaging paradigms. It supports the following types of communication:

- Synchronous request/response
- Asynchronous publish one-one
- Asynchronous publish one-many
- Asynchronous request/response (synchronous-to-asynchronous bridging)

## Message Brokering

A high performance message broker is a core component of Oracle Service Bus. It enables content-based routing of messages and data transformation. Oracle Service Bus message brokering capabilities are implemented with the following operational features:

- XQuery-based policies or callouts to external services for message routing
- Routing policies that apply to both point-to-point and one-to-many routing scenarios (publish). For publish, routing policies serve as subscription filters
- Routing table abstracted from proxy services, that enables modification of routes without having to re-configure proxy service definitions
- Identity-based routing, to classify clients into user-defined groups and apply routing policies based on these groups

## Conditional Routing

All routing logic pertaining to communications with a service end point is handled via the configured proxies. This frees service consumers from having to understand any of the complexities of communicating with back-end services. Decoupling the routing, transformation, security, and transport details from the service consumers and providers and placing them within configurable proxy services, provides for more flexible service integration.

Oracle Service Bus supports dynamic content-based routing of messages and run-time protocol selection. It facilitates these capabilities by allowing the configuration of proxy services with interfaces that are independent of the end-point business services. Using generic proxy templates, proxy services can be configured as message-flow definitions with routing logic that dynamically route messages to appropriate business services, based on message content.



## Message Transformation

Oracle Service Bus supports the following capabilities for the transformation or processing of messages:

- Validates incoming messages against schemas
- Selects a target service or services, based on the message content or message headers
- Transforms messages based on the target service
- Transforms messages based on XQuery or XSLT
- Supports transformations on both XML and MFL messages
- Message enrichment
- Supports callouts to Web services to gather additional data for transformation (for example, country code, full customer records, and so on)

## Service Callouts

Oracle Service Bus provides a service callout action that offers greater flexibility for more sophisticated message flows for complex dynamic-routing processing, or to perform message enrichment. The service callout action is used inside a message flow routing stage, to call on the destination service to perform some action on the message. The Service Callout functionality supports features such as RPC Encoding and URL replacement and offers extensibility of Oracle Service Bus capabilities by using Java Callouts and POJOs. For more information on service callouts, see [Constructing Service Callouts](#) in *Oracle Service Bus User Guide*.

## Database Lookup from Proxy Services

Oracle Service Bus provides a database lookup function using a new XQuery function in the Oracle XQuery engine. This can be used for message enrichment, for routing decisions or for customizing the behavior of a proxy service. Read-access to databases from proxy services is supported without requiring writing of a custom EJB or custom Java code and without the need for a separate database product like Oracle Data Service Integrator.

It is implemented using the `execute-sql()` function to make a JDBC call to a database to perform simple database reads. For more information, see “Accessing Databases using XQuery” in [Modeling Message Flow in Oracle Service Bus](#) in *Oracle Service Bus User Guide*.

## Native Transport for Oracle Data Service Integrator

Oracle Service Bus provides optimized transport for 1-way or 2-way communication for invoking services on Oracle Data Service Integrator data using a native transport (DSP). This allows a more efficient and flexible approach to accessing data services than exposing them as Web services via WebLogic Workshop and Java Web Services (JWS), and it supports security and identity propagation. For information on DSP transport, see the [Oracle Data Service Integrator and Oracle Service Bus Compatibility Matrix](#).

## Data Transformation Tools

Two data transformation tools are installed with Oracle Service Bus and Workshop for WebLogic—the Oracle XQuery Mapper plug-in for Eclipse 3.1 and Format Builder. Eclipse 3.1 and Format Builder are supported on Windows platforms only.

## Interoperability

For information about Oracle Service Bus interoperability, including information about support for compliance with messaging standards including SOAP, HTTP, JMS, SMTP/POP/IMAP, FTP, SSL, XML, XML Schema, WSDL, WSRP, and WS-Security, see [Oracle Service Bus Product Support Information](#).

## EJB Transport

Business services and Proxy services in Oracle Service Bus can be designed to use the EJB transport. The EJB transport is fully integrated into the Oracle Service Bus configuration, management, monitoring, and test consoles. Business services built with the EJB transport can be used for Publish, Service Callout, and service invocations.

An EJB can be exposed as a Web service, without the need for tools or the modification of the legacy code on the application server that hosts the EJB. The EJB transport provides the following capabilities:

- **Transactional Integrity:** EJB Transport can call business services in the context of a global transaction and can also suspend or start a global transaction before invoking an EJB.
- **Security Propagation:** An incoming SOAP over HTTP request to Oracle Service Bus is authenticated by Oracle Service Bus and the authenticated subject can then be propagated to the EJB server.

- **HTTP Tunneling and Encrypted Communication:** EJBs that are behind a fire wall can be accessed with HTTP tunneling. SSL can be used to encrypt all of the communications with the EJB Server.
- **JNDI Provider:** EJB transport leverages the JNDI provider, which can be reused by multiple EJB business services. This provides a centralized way for administrators to manage remote EJB server configurations.
- **High Performance Caching:** EJB transport is built on high performance cache and allows the reuse of established connections and minimizes EJB stubs lookups.
- **Failover and Load Balancing:** EJB transport can take advantage of scenarios in which the same EJB is deployed in multiple domains or on a cluster for load balancing or failover or both.
- **Advanced XML to Java Binding Capabilities:** EJB transport leverages the WebLogic Server JAX-RPC stack to perform Java to XML bindings.
- **Intelligent Retries:** EJB transport makes retry decisions based on the nature of the failure that can occur during the invocation of an EJB.

## Service Security

Oracle Service Bus supports open industry standards for ensuring the integrity and privacy of communications and to ensure that only authorized users can access resources in an Oracle Service Bus domain. It uses the underlying WebLogic security framework as building blocks for its security services. The WebLogic security framework divides the work of securing a domain into several components (providers), such as authentication, authorization, credential mapping, and auditing.

## Message Security Features

Oracle Service Bus provides the following security features:

- Authentication, encryption and decryption, and digital signatures as defined in the Web Services Security (WS-Security) specification
- Uses SSL to support traditional transport-level security for HTTP and JMS transport protocols
- One-way and two-way certificate based authentication
- HTTP basic authentication

- Encrypt and export of resources (such as service accounts, proxy service providers, UDDI registries, SMTP providers, and JNDI providers) that contain username and passwords
- Create service accounts and proxy service providers within a session, and add the user name, password, and credential alias binding within the same session.
- Configure a service account to pass through user ID and password credentials or map the user to a new user ID and password supplied to a business service
- Client-specified custom authentication credentials for both transport- and message-level inbound requests

The Oracle Service Bus security model includes the following:

- Inbound Security
- Outbound Security
- Options for Identity Propagation
- Administrative Security
- Configuring the WebLogic Security Framework
- Supported Standards and Security Providers

For more information on the above features of Oracle Service Bus security model, see [“Oracle Service Bus Security” on page 3-13](#).

For more information on Oracle Service Bus supported standards, see [Supported Standards and Security Providers](#) in *Oracle Service Bus Security Guide*.

## Service Composition

### Change Center

Oracle Service Bus Console Change Center is key to making configuration changes inside the service bus. The Change Center has the unique ability to lock its current configuration while changes are being made, letting the service bus continue to receive and process requests for services while configuration changes are being made in the console.

Changes being made to the configuration do not affect the current system configuration until they are “activated”. The service bus uses the new service and resource configuration when changes are activated. This way, ongoing changes can be made without disrupting services.

For more information on Change Center, see [Using the Change Center](#) in *Using the Oracle Service Bus Console*.

## Test Console

Oracle Service Bus built-in test console is a browser-based test environment used to validate resources and inline XQuery expressions used in the message flow. It is an extension of the Oracle Service Bus Console. Using the test console, it is possible to configure the test object (proxy service, business service, XQuery, XSLT, MFL resource), execute the test, and view test results. It allows message flow tracing when testing a service, to examine the state of the message at specific trace points.

Design time testing helps isolate design problems before deploying a configuration to a production environment. The test console can test specific parts of a system in isolation and it can test a system as a unit. The test console can be invoked in a number of ways in the Oracle Service Bus Console, from:

- The Project Explorer
- The Resource Browser
- The XQuery Editor

For more information, see [Using the Test Console](#) in the *Oracle Service Bus User Guide*.

## Resource Management

Oracle Service Bus provides the following resource management capabilities:

- Stores information about services, schemas, transformations, WSDLs (Web Service Definition Language), and WS Policies
- Provides centralized management and distributed access to resources and services
- Allows browsing of services registered in Oracle Service Bus and import of resources from WebLogic Workshop or other applications
- Allows the propagation of configuration data from environment to environment (for example, from a development domain to a test domain to a production domain). The system allows environment specific settings to be overridden during import.
- Allows for better synchronization and notification capabilities.

## Resource Customization

Oracle Service Bus provides a number of APIs for customization of service definitions, WSDLs, schemas, XQueries and other design-time resources through programmatic interfaces. The supporting APIs allow loading ZIP files containing resources, in addition to moving, renaming, cloning, or deleting resources, folders and projects. For more information, see [Oracle Service Bus APIs](#) in the *Oracle Service Bus User Guide*.

## UDDI Service Registry

Proxy services developed in Oracle Service Bus can be published to a UDDI registry. Oracle Service Bus can interact with any UDDI 3.0 compliant registry including Oracle Service Registry.

Service definitions in Oracle Service Bus can be synchronized (both ways) with those in UDDI. Services can be auto-published to UDDI after they are created or changed within Oracle Service Bus and business service definitions can be imported from UDDI.

Business services in Oracle Service Bus are also automatically updated (with no human intervention) when the original service is changed in UDDI. Alternatively, the Oracle Service Bus Console can be configured to prompt users for approval for synchronization when a service changes in the UDDI registry. For more information about UDDI registries, see [UDDI](#) in *Oracle Service Bus User Guide* and the [Oracle Service Registry](#) documentation.

## Error Handling

Oracle Service Bus supports the following error handling capabilities:

- Configure system to format and send error messages, and return messages for consumers of services who expect a synchronous response
- Configure error handling logic for pipeline stages, entire pipeline, and for proxy services
- Generate alerts based on message context in a pipeline, to send to an alert destination.

# Service Management

## Service Logging and Monitoring

Oracle Service Bus allows the following capabilities for auditing and monitoring services:

- Gather statistics about message invocations, errors, performance characteristics, messages passed and SLA violations
- Send SLA-based alerts as SNMP traps, enabling integration with third-party ESM solutions
- Support for logging selected parts of messages for both systems operations and business auditing purposes
- Search capabilities by extracting key information from a message and use as it as a search index.

The Oracle Service Bus Console provides a cluster-wide view of service status and statistics. Both business services and Oracle Service Bus proxy services are monitored, as are response times, message counts, and error counts.

JMX Monitoring API provided as a polling interface for the retrieval of metrics. This API enables integration with management partners and enables customers who have their own monitoring consoles to display metrics that can be used for performance analysis.

## Custom Operations Console

The Oracle Service Bus Console supports tasks performed by users in the operator (IntegrationOperator) role. It provides operational functions and settings that allow users to easily search for resources using the new SMarT Search functionality, monitor SLA alerts, pipeline alerts, logs, reports, turn tracing on and off, and to enable and disable services.

Users can readily distinguish between SLA and pipeline alerts since the metrics reported for each are distinguished on the console and via the JMX monitoring APIs. Service-level flags and global flags help control alerting (SLA & pipeline), reporting, and logging. Operators have privileges to edit operational settings, create new SLA alert rules, and create and edit alert destination resources.

For more information on console operational tasks, see [Oracle Service Bus Operations Guide](#).

## Service Versioning

Oracle Service Bus provides the ability to deploy new versions of services and allows you to have multiple versions of message resources such as WSDLs and schemas. Versions can include changes to the WSDL, the message schema, the headers, and the security parameters.

## Service Level Agreements

In Oracle Service Bus, monitoring statistics are gathered locally and aggregated centrally. SLA rules are run against aggregated data and the system raises alerts, following which services can be enabled or disabled. Administrators can set service level agreements (SLAs) on the following attributes of proxy services:

- Average processing time of a service
- Processing volume
- Number of errors, security violations, and schema validation errors
- Administrators can configure alerts for SLA rule violations

# Oracle Service Bus Deployment

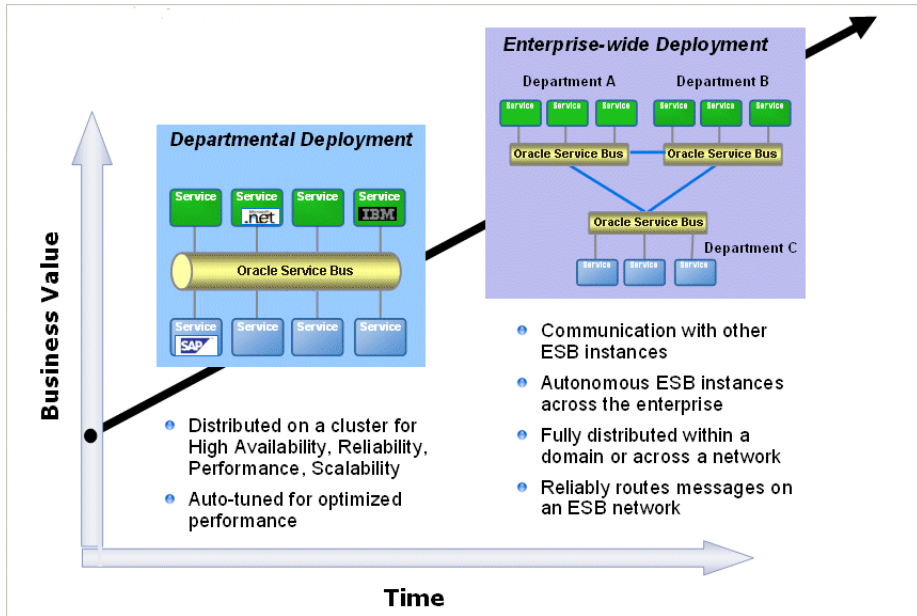
## Deployment Topology

Oracle Service Bus is designed to centrally manage and control many distributed service endpoints. Oracle Service Bus runs on WebLogic Server version 9.0 or greater, and can be deployed in the following configurations:

- On a single server that also serves as the administration server.
- On a cluster of servers.



Figure 2-6 Oracle Service Bus Deployments



Using Oracle Service Bus you can configure autonomous ESB instances across the enterprise. These instances have their own sets of configuration artifacts such as services and transformations. Such deployments typically map to various IT departments within an organization. Communication between different departments is achieved through a federated network of ESBs, which talk to each other, often through firewalls. For more information on Oracle Service Bus deployment, see [Oracle Service Bus Deployment Guide](#).

## Distributed Configurations for Large-Scale Deployments

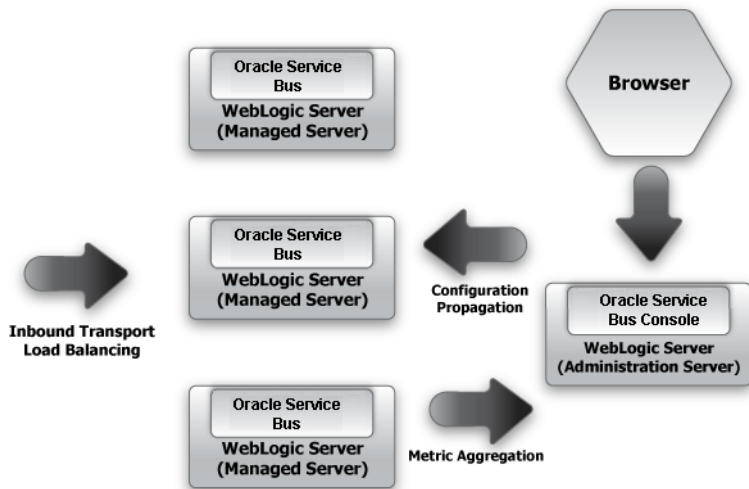
Oracle Service Bus enables management and coordination of many distributed service endpoints, thereby providing centralization in the enterprise. It is possible to horizontally scale heterogeneous Oracle Service Bus hubs by clustering the underlying WebLogic Server, to create a distributed Oracle Service Bus network.

A *cluster* consists of a set of clustered managed servers that perform message processing. A domain can have only one cluster with Oracle Service Bus deployed to it. This cluster can host other applications in addition to Oracle Service Bus. There is one administration server in every clustered domain. For more information on clustering, see [Understanding Oracle Service Bus Clusters](#) in [Oracle Service Bus Deployment Guide](#).

In this case, a central deployment of Oracle Service Bus is used for governance and coordination, and every ESB in the network communicates through the central ESB. It propagates configuration and metadata automatically to the managed servers for fast local retrieval, and it automatically collects monitoring metrics from all the managed servers for aggregation and displays them on the Oracle Service Bus Console.

The following figure shows the flow of message data in a basic Oracle Service Bus cluster topology.

**Figure 2-7 Clustering and High Availability**



Oracle Service Bus enables reliable and guaranteed messaging through the federated network through JMS store and forward. The dynamic routing capability also simplifies configuration of such a network. Spreading the messaging load homogeneously over the clustered servers prevents bottlenecks in the system.

## Development, Staging, and Production Domains

Oracle Service Bus supports best practices for change management in enterprise systems by configuration of resources and services in a controlled environment. It allows export of system configurations into separate staging domains for testing and final preparation for promotion into a production domain. Java programs or scripts can be used to automate deploying an application or for moving a configuration from staging to production.

The Oracle Service Bus Console has options for numerous deployment customization options. An extended list of environment variables allows settings to be preserved or tailored when moving from one environment to another. For more information, see “Finding and Replacing Environment Values” and “Creating Customization Files” in [Customization](#) in *Using the Oracle Service Bus Console*.

## Configuration Metadata Export and Import

An important part of large scale development is the ability to develop, test, stage, and deploy resources to a production system. Oracle Service Bus Console enables Oracle Service Bus resource configurations to be saved as metadata and exported in JAR files to other Oracle Service Bus domains. This functionality supports an orderly promotion process of Oracle Service Bus resource configurations from staging and test environments into production and minimizes the expertise, time, and resources needed to achieve various deployment scenarios.

In addition to exporting and importing resources, it is also permitted to export and import entire projects. Using the features of existing source code control system in conjunction with the configuration JAR files, provides version and change management for Oracle Service Bus configurations.

### Metadata Export

The Export feature provides the ability to export existing configurations using a JAR file, to another Oracle Service Bus domain. This capability allows system configuration to be propagated from one instance of Oracle Service Bus to another instance by exporting all, or a subset of, the resources deployed in the source Oracle Service Bus domain. There are no restrictions on what can be exported. One or more projects, or select resources from one or more projects can be exported.

The Oracle Service Bus Console also allows export of a resource and all the other resources on which it depends, using the dependency tracking feature. It is necessary to be working outside a session to export configurations. Only configurations that have been activated (that is, deployed to run time) can be exported.

There are two types of operational values: Global Operational Settings and operational values for proxy and business services. Global Operational Settings is a resource located in the System project folder and can be exported like any other resource. It is possible to preserve operational settings in the importing domain from being overwritten during import. This is achieved by specifying the 'Preserve Operational Values' setting. If 'Preserve Operational Values' is not specified, the values from the JAR file being imported are set in the domain.

## Metadata Import

The Import feature provides the ability to import resource configurations into a session on another Oracle Service Bus domain. To use the Import feature, it is necessary to be working in the session into which the configuration JAR file is to be imported. Many configuration updates and import of multiple JAR files is permitted in a single session. It is also possible to import only a subset of the exported data.

Oracle Service Bus provides the ability to reconfigure environment-specific elements as necessary to meet the requirements of the importing domain, using the Change Center in the Oracle Service Bus Console. Using this customization feature, imported resources can be tailored for the new domain before activating them.

It supports the global change of environment-specific attributes for resources, using the import functionality along with the find and replace feature. This facilitates changing of many similar environment values in a convenient way. It is not meant to replace a more careful tuning of configuration that may be required by complex deployment scenarios. For more information, see [Best Practices for Deploying Oracle Service Bus Resources](#).

For information on how to export and import configuration metadata using the Oracle Service Bus Console, see [Customization](#) in the *Using the Oracle Service Bus Console*. For information on how to modify configurations for new environments using the Oracle Service Bus Console Change Center, see [Using the Change Center](#) in the *Using the Oracle Service Bus Console*.

## Scripting Support

Though all Oracle Service Bus configuration and deployment can be performed using the Oracle Service Bus Console, the **WebLogic Server Scripting Tool (WLST)** can be used to automate deployment tasks. For information about WLST, see [WLST Command and Variable Reference](#) in *WebLogic Scripting Tool*.

## Related Topics

- [Modeling Message Flows in Oracle Service Bus](#) in the *Oracle Service Bus User Guide*.
- [Oracle Service Bus APIs](#) in *Oracle Service Bus User Guide*.
- [Supported Standards and Security Providers](#) in *Oracle Service Bus Security Guide*
- [Oracle Service Bus Deployment Guide](#)
- [Best Practices for Deploying Oracle Service Bus Resources](#)
- [Understanding Oracle Service Bus Clusters](#) in *Oracle Service Bus Deployment Guide*.



# Service Integration

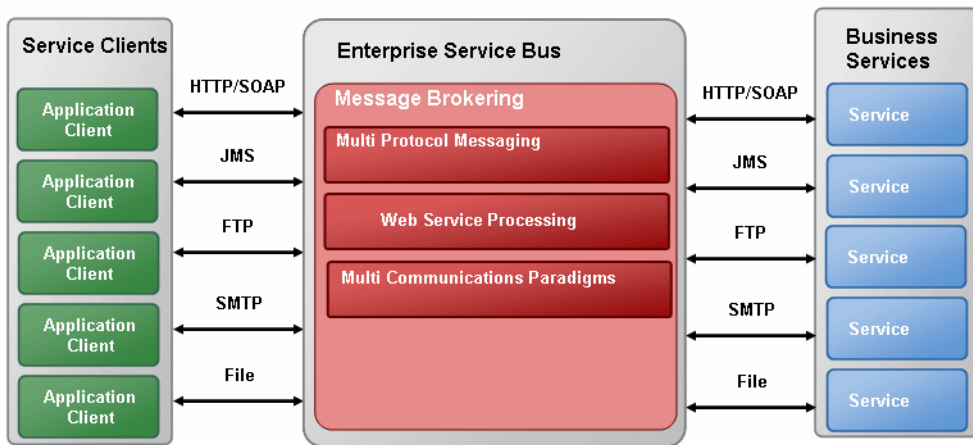
This section discusses Oracle Service Bus service integration and message brokering concepts and its capabilities which support high-speed reliable service mediation and provisioning. It is intended for IT deployment specialists who are responsible for configuring services in an SOA. This section includes the following topics:

- [“Message Brokering” on page 3-2](#)
- [“Oracle Service Bus Resources” on page 3-9](#)
- [“Oracle Service Bus Security” on page 3-13](#)

## Message Brokering

Oracle Service Bus provides capabilities to extend and integrate applications into enterprise-class services (web and legacy). It also provides facilities for mediating and exposing services for reuse, through intelligent brokering functionality.

**Figure 3-1 Message Brokering in Oracle Service Bus**



Using Oracle Service Bus, service providers and clients exchange messages with an intermediary proxy service instead of directly with each other, eliminating complexities resulting from heterogeneous communication protocols and messaging formats. It leverages the following operational capabilities to provide high-speed and reliable service mediation:

- Support for multiple Web Service transports - HTTP/SOAP, WS-I, WS-Security, WS-Policy, WS-Addressing, SOAP v1.2 and v1.1
- Support for traditional messaging transports - JMS, MQ, EJB/RMI, Tuxedo, FTP, SMTP, File, EJB/RMI on WebSphere
- Facility for database lookup
- Support for native transport for Oracle Data Service Integrator.
- Support for enterprise-specific custom transports creation, using the Oracle Service Bus Transport SDK



- Facility to create and configure generic proxy-services that can accept any SOAP or XML message, using generic proxy service templates
- Interoperability with web service integration technologies including .NET, Tibco EMS, IBM MQ, IBM WebSphere, Apache Axis, Cyclone B2B Interchange, and iWay 5.5 adapters

Its industry-leading support for Web services, traditional messaging protocols, and compatibility with legacy and proprietary integration technologies, make Oracle Service Bus an ideal service integration and message brokering solution. For more details on The following topics describe concepts related to service integration and message brokering using Oracle Service Bus.

## Services

In Oracle Service Bus, service integration relationships are implemented dynamically by configuring policies and proxy services. Both, proxy services and business services invoked by proxy services, are modeled as services that have the following attributes:

- A set of concrete interfaces called *ports* (also called an *endpoint*), each with a transport address and associated configuration. A set of ports constitutes load balancing and failover alternatives for a business service. A proxy service has only a single port.
- A single optional abstract interface which is the definition of the structure of message parts in the interface, optionally broken down by operations. Operations are equivalent to methods of a Java interface.
- A single binding that defines the packaging of message parts in the abstract interface to a concrete message.
- Policies on Web Service Security (WS-Security).

## Service Types

Oracle Service Bus supports varied service types ranging from conventional Web services (using XML or SOAP bindings in WSDLs) to non-XML (generic) services. The service type is selected by the individual doing the service registration when the business and proxy services are created, and it defines the protocols that can be used to communicate with the service end point. Oracle Service Bus service types include:

- **SOAP services:** SOAP messages are constructed by wrapping the contents of the header and body variables inside a <soap:Envelope> element. If the body variable contains a piece of reference XML, it is sent as is. In other words, the referenced content is not substituted into the message. If attachments are defined in the attachments variable, a MIME package is created from the main message and the attachment data. Content handling for each attachment part is similar to how it is handled for messaging services.
- **XML services (non-SOAP):** Messages to XML-based services are XML, but can be of any type the proxy service configuration allows. In messages that include attachments, their content is a MIME package that includes the primary XML payload as one of its parts—typically the first part or the one identified by the top-level content-type header.
- **Messaging services:** Messaging services are those that can receive messages of one data type and respond with messages of a different data type. Supported data types include XML, Message Format Language (MFL), text, untyped, binary and attachments where interface is not described by WSDL.

Oracle Service Bus supports request and response as well as one-way paradigms, for both the HTTP and the JMS asynchronous transport protocols. If the underlying transport supports ordered delivery of messages, Oracle Service Bus also extends the same support.

## Transport Protocols

Oracle Service Bus supports the following service transport protocols:

- EJB/RMI (Business Services only)
- Email (POP/SMTP/IMAP)
- File
- (S)FTP
- HTTP(S)
- JCA
- JMS (including MQ using JMS, and JMS/XA)
- Local (Oracle Proprietary for inter-ESB communication)
- MQ (WebSphere MQ)
- SB (RMI support)
- Tuxedo (Oracle Tuxedo)
- WS (WSRM, Web Services Reliable Messaging)

The service type selected defines the protocol to be used for communicating with the service end point. Table 1 shows the service types and supported transports :

**Table 3-1 Supported Service Types and Transports**

Service Type	Transport Protocols
SOAP WSDL or XMLnic	HTTP(S), JCA, JMS <sup>1</sup> , Local, SB, WS
SOAP (no WSDL)	HTTP(S), JMS <sup>1</sup> , Local, SB
XML (no WSDL) <sup>2</sup>	Email, File, FTP, HTTP(S), JMS, Local, MQ, SB, SFTP, Tuxedo
Messaging Type (Binary, Text, MFL, XML)	E-mail, File, FTP, HTTP(S), JMS, Local, MQ, SFTP, Tuxedo

1. JMS request and JMS response are not supported if WS-Security is enabled

2. HTTP GET is only supported for XML with no WSDL.

Oracle Service Bus also provides a Transport SDK to enable addition of native custom connectivity options.

For information on how to configure transport for a proxy service using the Oracle Service Bus Console, see “Adding a Proxy Service” in [Proxy Services](#) in *Using the Oracle Service Bus Console*.

For information on how to configure transport for a business service using the Oracle Service Bus Console, see “Adding a Business Service” in [Business Services](#) in *Using the Oracle Service Bus Console*.

## Service Interfaces

Oracle Service Bus relies on WSDLs for the formal description of Web services. For Web services, a WSDL describes what the Web Service’s interface is, where it resides, and how to invoke it. Oracle Service Bus defines proxy services and business services in terms of two WSDL entities:

- The abstract WSDL interface, which defines the operations in that interface and the types of message parts in the operation signature
- The binding WSDL interface, which defines the binding of the message parts to the message (packaging), and the binding of the message to the transport

WSDLs can be imported into the WSDL repository using the Oracle Service Bus Console. The Oracle Service Bus Console can also be used to resolve the references in the WSDLs, to ensure all schemas and WSDLs are linked correctly. After WSDLs are stored in the repository, they are available for use when adding proxy services and business services. Oracle Service Bus uses its own representation of the interface for messaging services.

For information on how to import and resolve WSDLs using the Oracle Service Bus Console, see “Adding a New WSDL” in [WSDLs](#) in *Using the Oracle Service Bus Console*.

## Messaging Models

Oracle Service Bus accommodates multiple messaging paradigms and supports the following types of communication:

- Synchronous request/response
- Asynchronous publish one-one
- Asynchronous publish one-many

- Asynchronous request/response (synchronous-to-asynchronous bridging).

In sync-async bridging, a synchronous client issues a request to an asynchronous provider. For this pattern, Oracle Service Bus provides the capability to publish a message on one JMS queue and configure a second JMS queue for the response, with a timeout value for listening for the response. This type of service appears as a synchronous service to the service consumer. Using asynchronous request/response messages has these advantages:

- No blocking by the request thread, removing thread management issues that can occur when numerous blocking request/response invocations are made.
- More reliable messaging

## Message Formats

Oracle Service Bus supports the following message formats:

- E-mail with or without attachments
- JMS with headers
- MFL (Message Format Language)
- Raw Data. (Raw data is opaque non-XML data with no known schema (no MFL file))
- Text
- SOAP and SOAP with attachments (SOAP described or not described by a WSDL)
- XML and XML with attachments (XML described or not described by a WSDL or a schema)

## Message Context

All messages sent to and received by the proxy service are defined internally in the proxy service by a set of properties that holds the message data and meta-data related to that message. This set of properties is known as the Message Context (context) and is implemented using Context Variables. It is defined by an XML schema. Each Context Variable relates to a different property. Some Context Variables are predefined and others are user defined. The heart of the proxy service is the *Message context*. For a complete description of the Message Context and context variables used in the message flow, see [Message Context](#) in *Oracle Service Bus User Guide*.

Predefined context variables contain information about the message, transport headers, security principals, metadata for the current proxy service and metadata for the primary routing and publish services invoked by the proxy service. You typically use an XQuery expression to manipulate context variables in a message flow. They can also be modified using transformation and in-place update actions.

The message related context variables `$header`, `$body`, and `$attachments` represent the canonical format of the message in the message flow. These are wrapper variables that contain the SOAP header elements, the SOAP body element, and the MIME attachments, respectively. The context gives the impression that all messages are SOAP messages and non-SOAP messages are mapped to this paradigm. The following table lists the mappings for each message type.

**Table 3-2 Message mappings**

Message Type	What Happens
<b>XML</b>	The <code>Body</code> element in <code>\$body</code> contains the XML document. Attachments are in <code>\$attachments</code> .
<b>binary</b>	The <code>Body</code> element in <code>\$body</code> contains a reference XML document. Attachments are in <code>\$attachments</code> .
<b>MFL</b>	The document is transparently converted from and to XML, and appears as an XML document in the <code>Body</code> element in <code>\$body</code> . Attachments are in <code>\$attachments</code> .
<b>text</b>	The <code>Body</code> element in <code>\$body</code> contains the text. Attachments are in <code>\$attachments</code> .
<b>File, FTP, and E-mail</b>	In the case of pass-by-reference documents, a reference XML document in the <code>Body</code> element in <code>\$body</code> refers to the URI of the document stored in the file system by the transport. Attachments are in <code>\$attachments</code> .
<b>SOAP</b>	The <code>Body</code> element in <code>\$body</code> contains the SOAP body. The <code>Header</code> element in <code>\$header</code> contains the SOAP header. Attachments are in <code>\$attachments</code> .

In the case of attachments, `$attachments` contains the following for each attachment:

- attachment, if the attachment is XML
- a reference XML, if the attachment is binary
- text, if the attachment is text

## Content Types

To support interoperability with heterogeneous end points, Oracle Service Bus lets service configurations control the content type, JMS type, and encoding used. It does not make assumptions about what the external client or service needs, but instead uses the service-definition information that has been configured for this purpose. Oracle Service Bus derives the content type for outbound messages from the service type and interface and uses the following specifications:

- XML or SOAP (with or without a WSDL), the content type is text/XML
- Messaging and the interface is MFL or binary, the content type is binary/octet-stream
- Messaging and the interface is text, the content type is text/plain
- Messaging and the interface is XML, the content type is text/XML.

The content type can be overridden in the outbound context variable (\$outbound) for proxy services invoking a service, and in the inbound context variable (\$inbound) for a proxy service response. Additionally, there is a JMS type (byte or text) which can be configured when the service is defined in the Administration Console. Encoding is also explicitly configured in the service definition for all outbound messages.

## Oracle Service Bus Resources

Oracle Service Bus resources are reusable definitions or descriptions of entities that typically include metadata for those entities. Resources can be used by multiple services and provide standardized definitions or descriptions for use across an enterprise or department.

Resources and services in Oracle Service Bus are grouped into a set of *projects*, each with a hierarchy of folders. Organizing resources and services into projects eliminates name conflicts and provides a convenient way to organize resources and services by business categories and search for them.

This section discusses the following includes the following Oracle Service Bus resources:

- [“Schemas and Data Types” on page 3-10](#)

- [“Transformation Maps” on page 3-11](#)
- [“JARs” on page 3-11](#)
- [“WSDLs” on page 3-11](#)
- [“Proxy Services” on page 3-12](#)
- [“Service Accounts” on page 3-14](#)
- [“Proxy Service Providers” on page 3-12](#)
- [“Alert Destinations” on page 3-12](#)
- [“JNDI Providers” on page 3-13](#)
- [“SMTP Servers” on page 3-13](#)

## Schemas and Data Types

Schemas describe types for primitive or structured data. XML Schemas are an XML vocabulary that describe the rules that XML business data must follow. XML Schemas specify the structure of documents, and the data type of each element and attribute contained in the document. XML schemas can import or include other XML schemas. For information on how to create schemas using the Oracle Service Bus Console, see “Adding a New Schema” in [XML Schemas](#) in *Using the Oracle Service Bus Console*.

Oracle Service Bus uses a metadata language called Message Format Language (MFL) to describe the structure of typed non-XML data. The Oracle Format Builder tool creates and maintains metadata as a data file called an MFL document. For information on how to create MFL documents, see the [Format Builder Online Help](#).

## Type System

Oracle Service Bus has a built-in type system that is available for use at design time. When creating an XQuery expression in a condition, in-place update action, or transformation, the variable can be declared to be of a given type in an editor to assist in easily creating the XQuery. The types can be the following:

- XML schema types or elements
- WSDL types or elements
- MFL types



## Transformation Maps

Transformation maps describe the mapping between two disparate data types of different source and destination services. Oracle Service Bus supports data mapping using either XQuery or the eXtensible Stylesheet Language Transformation (XSLT) standard. In addition, MFL described data is automatically converted to the equivalent XML for transformation with XQuery or XSLT. The resulting XML is automatically converted to MFL if the target service requires it.

## JARs

A JAR (Java ARchive) is a zipped file that contains a set of Java classes. It is used to store compiled Java classes and associated metadata that can constitute a program. A JAR acts like a callable program library for Java code elements (so that a single compilation link provides access to multiple elements, rather than requiring bindings for each element individually).

JAR files can be registered as reusable Oracle Service Bus resources. They are used in Java callout actions that provide a Java exit mechanism, EJB-based business services, and Tuxedo-based business services. For more information on JAR resources, see [JARs](#) in *Using the Oracle Service Bus Console*.

## WSDLs

A WSDL (Web Service Definition Language) interface defines a service interface for a SOAP or XML service. It describes the abstract interface of a service including the operations in that interface and the types of message parts in the operation signature. It can also describe the binding of the message parts to the message (packaging), and the binding of the message to the transport. In addition a WSDL can describe the concrete interface of the service (for example, the transport URL).

For information on how to configure WSDLs using the Oracle Service Bus Console, see “Adding a New WSDL” in [WSDLs](#) in *Using the Oracle Service Bus Console*.

## Proxy Services

Proxy services are Oracle Service Bus definitions of intermediary Web services that are hosted locally on Oracle Service Bus used to route messages to multiple business services. They are generic services that can be configured with an interface that is independent of the business services. Using generic proxy templates, the proxy service can be defined in terms of an interface, message flow definitions, and policies, that dynamically route messages to appropriate business services, based on content-based routing logic. For more information on proxy templates, see [Chapter 5, “Service Composition”](#), topic “[Proxy Templates](#).”

A proxy service can also map message data into appropriate protocol formats required by the end-point business service, allowing for dynamic run-time protocol switching. If the proxy service requires credential-level validation, a *proxy service provider* can be created to manage security credentials, using the Oracle Service Bus Console.

For information on how to configure a proxy service using the Oracle Service Bus Console, see “Adding a Proxy Service” in [Proxy Services](#) in *Using the Oracle Service Bus Console*.

## Proxy Service Providers

Proxy Service Provider resources contain Public Key Infrastructure (PKI) credentials that proxy services use for decrypting inbound SOAP messages and for outbound authentication and digital signatures. PKI credentials are private keys paired with certificates that can be used for digital signatures and encryption (for Web Service Security) and for outbound SSL authentication. The certificate contains the public key that corresponds to the private key. For information on how to configure a proxy service provider using the Oracle Service Bus Console, see “Adding a Proxy Service Provider” in [Service Key Providers](#) in *Using the Oracle Service Bus Console*.

## Alert Destinations

Alert Destination resources capture a list of recipients that can receive alert notifications from the Oracle Service Bus. They are used by Alert actions configured in the message flow, and by SLA alert rules. An Alert destination could include one or more of the following types of destinations: Reporting Data stream, SNMP trap, E-mail, JMS queue, or JMS topic. In the case of E-mail and JMS destinations, a destination resource could include a list of E-mail addresses or JMS URIs, respectively. Alert Destinations can be re-used across alert configurations for services.

For more information on Alert Destination resources, see [Alert Destinations](#) in *Using the Oracle Service Bus Console*.

## JNDI Providers

JNDI Provider resources communication protocols and security credentials for accessing remote servers and can be reused from numerous proxy services. They are global resources that may be used in Alert Destination resources across projects within an Oracle Service Bus domain.

For more information on JNDI Providers, see “JNDI Providers” under topic [Global Resources](#), in *Using the Oracle Service Bus Console*.

## SMTP Servers

SMTP Server resources specify the address of SMTP servers corresponding to E-mail destinations, port numbers, and, if required, authentication credentials. They are global resources that are used in Alert Destination resources across projects in an Oracle Service Bus domain.

For more information on SMTP Server resources, see “SMTP Server” under [Global Resources](#) in *Using the Oracle Service Bus Console*.

# Oracle Service Bus Security

Oracle Service Bus uses the proven Oracle WebLogic Security Framework in Oracle WebLogic Server 9.0, as the building blocks for higher-level security services including authentication, identity assertion, authorization, role mapping, auditing, and credential mapping. Oracle WebLogic Server security is configured before the Console can be used to configure security.

The Console has predefined rules that simplify using the Oracle WebLogic Server security providers at several different levels in its operation. For more information on supported security levels, see section “[Security Levels](#)”.

For more information on Oracle Service Bus security functionality, see topic “[Understanding Oracle Service Bus Security](#)” in *Oracle Service Bus Security Guide*.

## WS-Policies

Web Services Policy (WS-Policy) is a standards-based framework for defining a Web service's security constraints and requirements. It expresses security constraints and requirements in a collection of XML statements called policies, each of which contains one or more assertions. In Oracle Service Bus, WS-Policy assertions are used to specify a Web service's requirements for digital signatures and encryption, along with the security algorithms and authentication mechanisms that it requires.

WS-Policy policies may be included directly in a WSDL document or included by reference, and a WSDL document may import other WSDL documents that contain or refer to WS-Policy policies. An XML file that contains these policies can be used by multiple proxy services or business services. The WebLogic Web Services runtime environment recognizes two types of WS-Policy statements:

- Concrete WS-Policy statements: specify the security tokens that are used for authentication, encryption, and digital signatures. These WS-Policy statements are created if the type of authentication required (such as using X.509 or SAML tokens), multiple private key and certificate pairs from the keystore used for encryption and digital signatures, are known at run-time.
- Abstract WS-Policy statements: that do not specify security tokens.

The Oracle Service Bus runtime environment determines which security token types an abstract policy will accept. For information on configuring the runtime environment, see [Oracle Service Bus WS-Policy Statements](#) in *Oracle Service Bus Security Guide*.

Policies are referenced by an URI, either embedded within a WSDL, an HTTP URI, or a policy URI (for example, `policy:myPolicy`). Policy URIs can reference in-built policies. For more information on WS-Policy, see [Oracle Service Bus Security Guide](#).

## Service Accounts

Service Account resources provide a user name and password that proxy services and business services use for outbound authentication or authentication to a local or remote resource, such as an FTP server or a JMS server. For example, if a business service is required to supply a user name and password for transport-level authentication with a Web Service, a service account can be created to specify the user name and password. The business service can then be configured to include the service-account credentials in its outbound requests. One service account can be used for multiple business services and proxy services. For more information on Service Account resources, see [Service Accounts](#) in *Using the Oracle Service Bus Console*.

## Security Levels

Oracle Service Bus provides the following types of security features:

- Authentication
- Identity assertion
- Authorization
- Auditing
- Credential mapping

The following topics discuss the security features available in the Oracle Service Bus security model.

### Inbound Security

Inbound Security ensures that Oracle Service Bus proxy services handle only the requests that come from authorized clients (by default, any anonymous or authenticated user can connect to a proxy service). It can also ensure that no unauthorized user has viewed or modified the data as it was sent from the client.

Proxy services can have two types of clients: service consumers and other proxy services.

Inbound security is set up when proxy services are created and is determined by varying security requirements. For outward-facing proxy services which receive requests from service consumers, strict security requirements such as two-way SSL over HTTPS are used. For proxy services that are guaranteed to receive requests only from other Oracle Service Bus proxy services, less secure protocols are used. If a proxy service uses public key infrastructure (PKI) technology for digital signatures, encryption, or SSL authentication, create a proxy service provider to provide private keys paired with certificates.

For each proxy service, the following inbound security checks can be configured:

- **Transport-level security:** applies security checks as part of establishing a connection between a client and a proxy service. The security requirements that you can impose through transport-level security depend on the protocol that you configure the proxy service to use. For information about configuring transport-level security for each supported protocol, see [Configuring Transport-Level Security](#). For more information, see [“Transport-Level Security” on page 3-18](#)

- **Custom Authentication:** for message-level security and client-specified custom authentication credentials for inbound transport- and message-level requests. The custom authentication credentials can be in the form of a custom token, or a username and password. For more information, see [“Custom Security Credentials” on page 3-19](#)
- **Message-level security:** for proxy services that are Web Services. This is part of the WS-Security specification. It applies security checks before processing a SOAP message or specific parts of a SOAP message. For more information, see [“Message-Level Security” on page 3-19](#)

## Outbound Security

Outbound security secures communication between a proxy service and a business service. Most of the tasks involve configuring proxy services to comply with the transport-level or message-level security requirements that business services specify. If a business service requires the use of PKI technology for digital signatures, or SSL authentication, a *proxy service provider* is created, which provides private keys paired with certificates. For more information, see [Proxy Service Providers](#) in *Using the Oracle Service Bus Console*.

## Options for Identity Propagation

Options for Identity Propagation allows for decision making when designing security for Oracle Service Bus, including how to propagate the identities that clients provide. Oracle Service Bus can be configured to do any of the following:

- Authenticate the credentials that clients provide
- Perform authorization checks
- Pass client credentials to business services unchanged
- Map client credentials to a different set of credentials that a business service can authenticate and authorize
- Bridge between security technologies

For detailed descriptions of these WebLogic Server security providers and WebLogic Server security architecture in general, see [Oracle WebLogic Server Security](#) documentation.

Oracle Service Bus security supports the WS-Policy specification. For more information on WS-Policy specification, see the Web Services Policy Framework (WS-Policy) and Web Services Policy Attachment (WS-PolicyAttachment) which is available at:

<http://specs.xmlsoap.org/ws/2004/09/policy/>

Using the Oracle Service Bus Console, it is possible to configure a service with security policies that apply to messages in its interface. A security policy can be specified for a service or for individual messages associated with the operations of a service. When a security policy is specified for a service, the policy applies to all messages sent to that service.

Oracle Service Bus enables you to use the WebLogic Server security providers at several different levels in its operation. The following levels of security are supported:

- [User Management](#)
- [Administrative Security](#)
- [Transport-Level Security](#)
- [Message-Level Security](#)

For more information on security levels, see [Oracle Service Bus Security Guide](#).

## User Management

Oracle Service Bus user management is built on the unified WebLogic Server security framework. This framework enables the Oracle Service Bus Console to support task-level authorization based on security policies associated with roles assigned to named groups or individual users. For more information on the WebLogic Server security framework, see the [Oracle WebLogic Server Security](#) documentation.

The Oracle Service Bus Console is used to manage Oracle Service Bus users, groups, and roles. For information on how to manage Oracle Service Bus users, groups, and roles using the Oracle Service Bus Console, see [Security Configuration](#) in the *Using the Oracle Service Bus Console*.

## Administrative Security

To give users access to administrative functions such as creating proxy services, they can be assigned to one of four security roles with pre-defined access privileges. A security role is an identity that can be dynamically conferred upon a user or group based on conditions that are evaluated at runtime. The access privileges for the Oracle Service Bus administrative security roles cannot be changed but the conditions under which a user or group is in one of the roles can be changed.

By default, the first user created for an Oracle Service Bus domain is a WebLogic Server Administrator. This user has full access to all Oracle Service Bus objects and functions, and can execute user management tasks to provide controlled access to Oracle Service Bus Console functionality.

The following is a list of default roles to which Oracle Service Bus users can be assigned:

- IntegrationAdmin
- IntegrationDeployer
- IntegrationMonitor
- IntegrationOperator

For information on configuring administrative security, see [Configuring Administrative Security](#) in *Oracle Service Bus Security Guide*.

For information on how to manage Oracle Service Bus users, groups, and roles using the Oracle Service Bus Console, see [Security Configuration](#) in the *Using the Oracle Service Bus Console*.

## Transport-Level Security

Oracle Service Bus supports transport-level confidentiality, message integrity, and client authentication for one-way requests or request/response transactions (from clients to Oracle Service Bus) over HTTPS. It allows HTTP(S) proxy services or business services to be configured to require one of the following types of client authentication:

- BASIC (username/password) client authentication
- CLIENT CERT (two-way SSL) client authentication
- No client authentication

When a proxy service is activated, Oracle Service Bus generates and deploys a thin Web application. Oracle Service Bus relies on WebLogic Server for server-side SSL support, including session management, client certificate validation and authentication, trust management and server SSL key/certificate manipulation.

Transport security for transports other than HTTP is supported in Oracle Service Bus as follows:

- For the E-mail and FTP transports, security is provided using credentials to connect to a FTP or E-mail server.
- For the file transport, security is provided using a login control to the machine on which the files are located.

For more information, see [Oracle Service Bus Security Guide](#).



## Message-Level Security

Oracle Service Bus supports OASIS Web Services Security (WSS) 1.0. For more information on the WSS specification, see the OASIS Web Services Security TC which is available at:

[http://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=wss](http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss)

WSS defines a framework for message confidentiality, integrity, and sender authentication for SOAP messages. Using WSS, Oracle Service Bus provides support for securing messages using digital signatures, encryption, or both. Though it is not a substitute for transport-level security, WSS is ideal for end-to-end message confidentiality and integrity.

It is more flexible than SSL since individual parts of the SOAP envelope can be signed, encrypted or both, while other parts are neither signed nor encrypted. This is a powerful feature when combined with the ability of Oracle Service Bus to make routing decisions and perform transformations on the data based on the message content. Oracle Service Bus currently supports WSS over HTTP/S and JMS.

## Custom Security Credentials

There are several ways to authenticate a client's identity in Oracle Service Bus—using Basic Authentication, client certificates (2-way SSL), and Web Service Security. Client credentials associated with a business service and a proxy service are managed directly using WebLogic Server. Client-specified custom authentication credentials for both transport- and message-level inbound requests are also supported. The custom authentication credentials can be in the form of tokens, or a username and password token combination.

Oracle Service Bus accepts and attempts to authenticate:

- A custom token passed to a proxy service in an HTTP header, SOAP header (for SOAP-based proxy services) or in the payload (for non-SOAP proxy services).
- A username and password token passed in a SOAP header (for SOAP based proxy services), or in the payload for non-SOAP proxy services.
- The custom authentication mechanisms work alone or in concert with the message-level security for Web services.

For more information on custom security credentials, see [Configuring Custom Authentication](#) in the *Oracle Service Bus Security Guide*.

## Related Topics

- [Oracle Service Bus Security FAQ](#)
- [Security Configuration](#) in the *Using the Oracle Service Bus Console*.
- [Oracle Service Bus WS-Policy Statements](#) in the *Oracle Service Bus Security Guide*.
- [Securing Oracle Service Bus in a Production Environment](#) in the *Oracle Service Bus Security Guide*.

# Service Configuration

This section discusses the service configuration and resource organization capabilities provided by Oracle Service Bus. It highlights features that support service discovery and change management. This section is intended for IT deployment specialists who are responsible for configuring services in an SOA. This section includes the following topics:

- [“Resource Organization” on page 4-1](#)
- [“Change Management” on page 4-3](#)
- [“Service Discovery” on page 4-8](#)

## Resource Organization

Oracle Service Bus has a robust resource configuration and organization framework for creating, organizing and configuring resources and ensuring semantic integrity between resource dependencies. It provides features to rapidly test, deploy, and, reverse resource configuration updates if required.

## Project Explorer

Oracle Service Bus has a built-in Project Explorer that allows logical grouping of Oracle Service Bus entities, allowing developers and administrators to better organize related parts of large development projects. For more information, see [Project Explorer](#) in *Using the Oracle Service Bus Console*.

## Projects and Folders

Oracle Service Bus resources can be organized into separate projects. Projects are non-hierarchical, disjointed, top-level grouping constructs. All resources (such as services, WS-Policies, WSDLs, XQuery transformations, etc.) reside in exactly one non-overlapping project.

Resources can be created directly under a project or be further organized into folders. Folders may be created inside projects or inside other folders and are similar to directories in a file system, with the project level being the root directory. Descriptions can be added to all projects and folders to further enhance navigation. The following figure shows the project and folder views in the Oracle Service Bus Console.

**Figure 4-1 Project Explorer View of Oracle Service Bus Projects and Folders**

The screenshot displays the Oracle Service Bus Console interface, specifically the Project Explorer view. It is divided into four main sections:

- Projects:** At the top, there is a section for creating new projects with a text input for "Enter New Project Name:" and an "Add Project" button. Below this is a table listing projects. The first project is "default" (blue), and the second is "MortgageBroker" (pink). An "Options" button is visible next to the "default" project.
- MortgageBroker Details:** This section provides details for the selected "MortgageBroker" project. It includes a "Description" field, "References: 0", and "Referenced By: 0". There is an "Edit Description" button.
- Folders:** Below the project details, there is a section for creating new folders with a text input for "Enter New Folder Name:" and an "Add Folder" button. Below this is a table listing folders: "BusinessServices" (blue), "ProxyServices" (pink), and "WSDL" (blue). Each folder has an "Options" button.
- Resources:** At the bottom, there is a section for creating new resources with a text input for "Create Resource:" and a dropdown menu for "Select Resource Type:". Below this is a table listing resources, with columns for "Name", "Resource Type", "Actions", and "Options".

Resources can be moved between projects or folders and can be renamed. Any Oracle Service Bus resource, project or folder can be cloned to create a copy of that resource with the specified

target identity. Cloning a project or folder copies all artifacts in the project or folder to a different location.

Resources that are located in one project can reference and use resources that are defined in other projects. Dependencies are preserved when resources are renamed and moved. All references to a renamed or moved resource are automatically adjusted. An additional capability of the Project Explorer is the ability to track which resources outside the current folder are referenced by resources residing in it. Viewing these references gives both the location of the referenced resource (in the format of <project name>/<folder name>/<resource name>) and the type of resource referenced. For more information about referenced resources, see [“Tracking Dependencies”](#).

## Resource Cache

Oracle Service Bus provides a number of capabilities to organize large numbers of resources in the resource cache. The resources in the Oracle Service Bus resource cache include WSDLs, XML Schemas, XQueries, XSLTs, MFLs, WS-Policies, Business Services, and Proxy Services. Oracle Service Bus relies on user-configured metadata for resources and services to determine how to process messages.

Oracle Service Bus is focused on supporting a set of trusted IT department specialists who manage the resources and services in the resource cache on behalf of the organizations they represent. All such users are defined as integration administrators or integration deployers and have full permissions to modify all the resources in the resource cache. Integration monitor users have full read access to the resource cache but cannot modify any resources. Typically, they are users who search or browse for resources or services. Integration operator users have full read access to the resource cache and can only change the operational characteristics of the services.

For more information about Oracle Service Bus users and roles, see [Security Configuration](#) in *Using the Oracle Service Bus Console*.

# Change Management

## Change Center

One of the most important features of Oracle Service Bus is the Change Center, which is key to making configuration changes inside the service bus. The Change Center has the unique ability to lock its current configuration while changes are being made. This lets the service bus continue to receive and process requests for services while configuration changes are being made in the console.

Additionally, changes being made to the configuration will not affect the current configuration until they are “activated.” Once this is done, the changes go live instantly and the service bus immediately uses the new configuration. This way, ongoing changes can be made without disrupting services. For more information on Change Center features, see

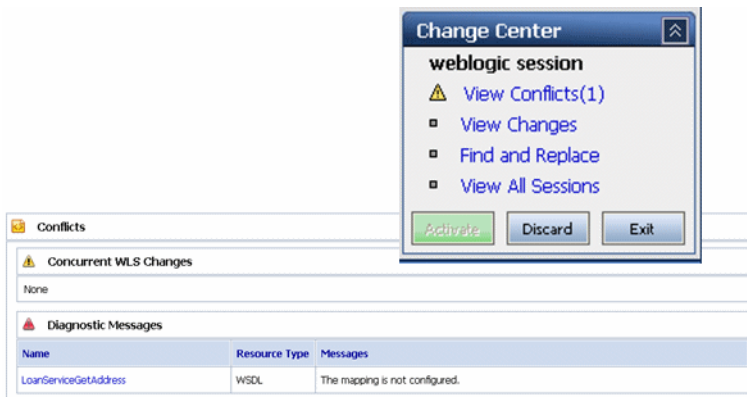
If activated changes cause unpredictable, undesirable events, the Change Center also provides the capability to undo any changes made for any session. Task Details provides information on which resource was changed, who changed it, and when. An entire session or individual changes within a session can be rolled back, enabling Oracle Service Bus to roll the affected configurations back to the prior state.

For more information on Change Center, see [Using the Change Center](#) in *Using the Oracle Service Bus Console*.

## Sessions

Before you make modifications to resources in Oracle Service Bus, you must create a session. All modifications are made using the Oracle Service Bus Console in a given session.

**Figure 4-2 Session Management in the Change Center**



A session can be considered a sandbox environment in which changes are kept private to the user making those changes. In other words, they are not visible to other concurrent users making modifications. Modifications made in a session are not deployed in the server until the session is activated. Only one session can be active at any time and should only log into the Oracle Service Bus Console through one browser.

To compare a resource modified in a given session against the resource that is already deployed to run time, you can temporarily exit the session, view the deployed resource, then reenter the session and view the changed resource. All resources are visible in the session. The view of all resources in a session is called the session view—it is a merged view of the unmodified deployed resources and the resources modified in the current session. Therefore, the session view at any point in time shows the configuration state if the session is activated at that point in time. The view of resources outside any session is the view of the deployed resources.

All individual session modifications, individual session activations, and undo operations for a session are performed in transactions to prevent data loss in the event of a failure. Sessions are persistent and long running—the restart of a server does not result in the loss of active sessions. This means that you can modify the configuration in a single session over a period of days (during which time the server can be stopped and restarted) if necessary. Each user has their own session, and can work in it independently without the need to lock other users out of the system.

You cannot activate a session if another user is already in the process of activating their session. If another user is activating a session when you try to activate your session, the Activate button will be disabled and you will have to wait until the other session is activated before you can activate your session. In certain circumstances, the Activate button may not be disabled if you did not refresh the page or if you are directly using MBeans. In this case, you will time out after a short while.

Administrators have permissions to access other user's sessions and view ongoing changes, make updates in those sessions, or discard them.

For more information, see [Using the Change Center](#) in *Using the Oracle Service Bus Console*.

## Concurrent Modifications

Sessions use an optimistic scheme for conflicts. When you activate a session, the changes you made to resources in that session become visible immediately in other sessions. If you deploy a changed resource that is open in another user's active session, the other user's session receives a message in the Change Center indicating that the deployed resource has changed in the run time since the user started modifications. The user of the active session can then:

- Discard the changes to the resource in the current session. That is, refresh the resource in the session with the newly deployed resource.
- Activate the current session, which results in the resource in the run time being overwritten with the current session's changes. This is the default behavior.

For more information, see [Using the Change Center](#) in *Using the Oracle Service Bus Console*.

## Tracking Configuration Changes

The system keeps a log of all users who activated a session along with any resource modified by the session and when it was modified. This provides the enterprise with auditing and tracking facilities in addition to a history of changes made to a particular resource or project. The log is visible in the Change Center in the Oracle Service Bus Console.

## Tracking Dependencies

A crucial part of managing a large number of resources is establishing and exploring dependencies between resources. For example, it is useful to identify the WSDL that a service implements, or the XQueries used by a message flow configuration. Oracle Service Bus provides this capability by automatically tracking the references between resources and creating a graph of the dependencies. In both session views and deployed views, the Oracle Service Bus Console displays for a given resource:

- The resources that it references
- The resources that it is referenced by

Also, for each project and folder, the Oracle Service Bus Console displays other resources outside the project or folder that reference resources in the selected project or folder. The Oracle Service Bus Console also displays the resources that a given project or folder references. This aids dependency tracking—you can easily navigate the dependency graph in the Oracle Service Bus Console by clicking on the names of the referenced resources.

You can use this functionality to identify the dependencies between departmental projects or between departmental projects and corporate-wide shared projects in the resource cache.

Dependencies are preserved when resources are renamed and moved. All references to a renamed or moved resource are automatically adjusted.

## Semantic Integrity

Oracle Service Bus protects the integrity of all resources in the session view. You can view a list of all current validation errors for all resources in the session view by clicking the View Conflicts link in the Change Center. Changes to a referenced resource can cause validation errors in any resources that reference it.

Oracle Service Bus allows you to create resources with most semantic errors. However, all such errors must be fixed before a session can be committed.



There are certain classes of validation errors that are never allowed. If you attempt to update a resource that has one of these disallowed validation errors, your update will fail. For example, where your configuration requires an XQuery, you cannot enter arbitrary text in place of the XQuery. If you try to do this, your update fails. The XQuery and XPath editors in the Oracle Service Bus Console provide a facility to validate your expressions. Click **Validate** to validate your XQuery and XPath expressions at design time. This reduces the possibility of run-time errors as a result of invalid configurations.

## Reversing Changes to Resources and Sessions

You can undo tasks that you have performed in your Oracle Service Bus configuration during your current session, and you can undo session activations outside of a session.

### Undoing Modifications to Resources

If you are working in a session, you can view a list of the modifications you have made in the session by accessing **View Changes** in the Change Center. You can undo specific tasks in the Change Center. An undo operation can result in objects becoming semantically invalid. For example, if a WSDL operation name change is undone, the proxy service routing to that operation on the service that uses that WSDL is semantically invalid. These validation errors are displayed immediately in the Change Center when you click the **View Conflicts** link.

Although you can undo tasks in any order (provided that individual undo operations result in valid data), the resulting configuration may be different depending on the order of undo. The undo operation sets the value of the resource to the value it had before the change to that resource. If the task being undone was one that created an object, there is no previous state to which an object can be returned—in other words, no object existed before this task was performed. Effectively, the undo operation deletes the new object from the session. In this case, errors occur for the objects that reference the one being deleted. You can view such errors on the **View Conflicts** page in the Change Center.

### Undoing Session Activations

When you are not working in a session, you can view a list of session activations by accessing **View Changes** in the Change Center. You can undo a session activation in the Change Center. When you undo a session, the session activation is undone and all the operations performed in the session are lost. The system does not allow you to undo a session activation if an error in the run time configuration would result from the undo operation.

For example, if you attempt to undo a deployment that removes an object that is being referenced by another object, that undo operation is disallowed. For more information, see “Undoing a Task” in [Using the Change Center](#) in *Using the Oracle Service Bus Console*.

## Service Discovery

### UDDI Registry

Universal Description, Discovery and Integration (UDDI) registries are used in an enterprise to share Web services. Using UDDI services helps companies organize and catalog these Web services for sharing and reuse in the enterprise or with trusted external partners.

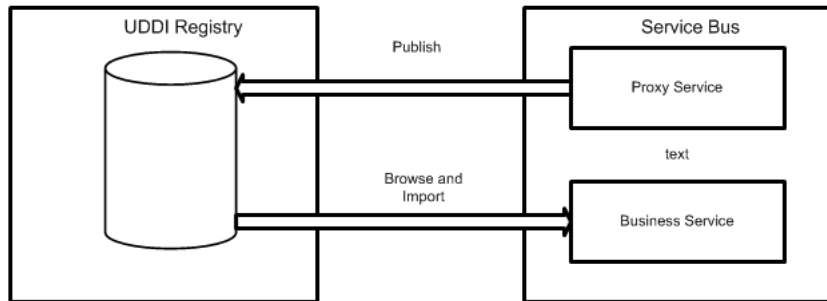
A UDDI registry service for Web services is defined by the UDDI specification available at:

<http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3>

UDDI registries are based on this specification, which provides details on how to publish and locate information about Web services using UDDI. The specification does not define run-time aspects of the services (it is only a directory of the services). UDDI provides a framework in which to classify your business, its services, and the technical details about the services you want to expose.

Publishing a service to a registry requires knowledge of the service type and the data structure representing that service in the registry. A registry entry has certain properties associated with it and these property types are defined when the registry is created. You can publish your service to a registry and make it available for other organizations to discover and use.

Proxy services developed in Oracle Service Bus can be published to a UDDI registry. Oracle Service Bus can interact with any UDDI 3.0 compliant registry including Oracle Service Registry.

**Figure 4-3 Figure 5-1 Oracle Service Bus integration with UDDI**

## Advantages of a UDDI Registry

UDDI offers several benefits to IT managers at both design time and run time, including increasing code reuse. UDDI also provides benefits to developers, including the following:

- UDDI improves infrastructure management by publishing information about proxy services to the registry and categorizes the services for discovery. Thus growing a portfolio of services making it easier to understand and manage relationships among services, component versioning, and dependencies.
- UDDI services can be imported from a registry to configure the parameters required to invoke the Web service and the necessary transport and security protocols.
- UDDI promotes the use of standards-based Web services and business services development in business applications and provides a link to a library of resources for Web services developers. This decreasing the development lifecycle and improves productivity. It also increases the prospect of interoperability between business applications by sharing standards-based resources.
- UDDI provides a user friendly interface for searching and discovering Web services. You can search on criteria specified by you.

## Oracle Service Registry

Oracle Service Registry is a version 3 compliant UDDI registry certified to work with Oracle Service Bus. It is not provided with Oracle Service Bus.

For information about Oracle Service Registry, see the product documentation at the following URL:

[http://download.oracle.com/docs/cd/E13173\\_01/alsr/docs21/index.html](http://download.oracle.com/docs/cd/E13173_01/alsr/docs21/index.html)

The Oracle Service Bus Console makes the Oracle Service Registry or any version 3 UDDI-compliant registry accessible and easy to use. In working with UDDI, Oracle Service Bus promotes the re-use of standards based Web services. In this way, Oracle Service Bus resources can be searched for and discovered and used by a wide and distributed audience. Web services and UDDI are all built on a set of standards, so re-use promotes the use of acceptable, tested Web services and application development standards across the enterprise. The Web services and interfaces can be catalogued by type, function, or classification so that they can be discovered and managed more easily.

Permissions in Oracle Service Registry were developed so that administrators can manage users' permissions in Oracle Service Registry and create views into the registry, specific to the needs of the different user types. User permissions set in Oracle Service Bus govern access to the registries, their content, and the functionality available to you.

### Publishing a Proxy Service to a UDDI Registry

You can use the Oracle Service Bus Console to publish proxy services to Oracle Service Registry. You can publish all proxy services to a UDDI registry—this includes the following service types: WSDL, Messaging, Any SOAP, and Any XML. For information on how to publish proxy services to a UDDI registry, see “Publishing a Proxy Service to a UDDI Registry” in **UDDI** in *Using the Oracle Service Bus Console*.

### Importing a Service from a UDDI Registry

You can import services from the registry as Oracle Service Bus business services. The service types supported are WSDL services with SOAP over HTTP binding and Oracle Service Bus proxy services (used primarily in multi-domain deployments). For information on how to import business services to Oracle Service Bus, see “Importing a Business Service from UDDI Registry” in **UDDI** in *Using the Oracle Service Bus Console*.

## Auto-Synchronization of Services With UDDI

Service definitions in Oracle Service Bus can be automatically synchronized (both ways) with those in UDDI. Services can be automatically published to a UDDI registry after they are created or changed within Oracle Service Bus and business service definitions can be imported from UDDI and automatically updated when the original service is changed in UDDI. Alternatively, you can configure the Oracle Service Bus Console to prompt you for approval for synchronization when a service changes in the UDDI registry. For more information, see "Configuring a UDDI Registry" in [UDDI](#) in *Using the Oracle Service Bus Console*.

## Related Topics

- [Oracle Service Registry](#)
- [UDDI](#) in *Oracle Service Bus User Guide*
- Technical Notes can be found at the following URL. Specifically, *Using WSDL in a UDDI Registry* is recommended:  
<http://www.oasis-open.org/committees/uddi-spec/doc/tns.htm>
- UDDI product and development tool information is available on the OASIS UDDI Solutions page at the following URL:  
<http://uddi.org/solutions.html>
- The UDDI specifications, which are available at the following URL:  
<http://www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm>

These specifications define:

- SOAP APIs that applications use to query and to publish information to a UDDI registry
  - XML Schema schemas of the registry data model and the SOAP message formats
  - WSDL definitions of the SOAP APIs
  - UDDI registry definitions (tModels) of various identifier and category systems that can be used to identify and categorize UDDI registrations
- Technical notes and best practice documents that help you deploy and use UDDI registries effectively are available on the OASIS UDDI Web site at the following URL:  
<http://uddi.org>

## Service Configuration

# Service Composition

This section discusses the service composition capabilities of Oracle Service Bus. It highlights operational features that enable service configuration and key concepts associated with message structures and message flow modeling. It is intended for integration-focused IT architects responsible for messaging and service oriented architectures (SOA) and service modelers or designers. This section includes the following topics:

- [“Dynamic Content-Based Routing” on page 5-1](#)
- [“Message Flow Modeling” on page 5-3](#)
- [“Transformations” on page 5-10](#)
- [“Error Handling” on page 5-12](#)

## Dynamic Content-Based Routing

Oracle Service Bus mediates service request and response messages between disparate heterogeneous service endpoints and intelligently routes messages between them. Content-based routing is a mediation capability supported by Oracle Service Bus based on conditional message processing and transformation capabilities. This routing capability allows loose coupling of SOA endpoints and is particularly useful and allows service enrichment and reuse by combining transformation and routing functions.

Oracle Service Bus performs dynamic message routing based on a message content, for cases when services or responses need to be directed to multiple destination service and in scenarios where different versions of a service have to be provisioned based upon business service requests.

Dynamic routing is useful when business requirements dictate that certain conditions of a request define where it should be processed. For example, a financial institution's request for a credit report on a customer may use any of several credit services based on where the customer or organization resides.

In dynamic routing, a message is analyzed using conditional checks in conditional branching statements, to retrieve the value of a data element or multiple data elements that determine the routing logic. Different business service destinations are assigned to different value combinations resulting from this conditional check. The message is dynamically routed to one of multiple destination business services based on the data element value. Transformations may be applied to the response message going to one or more of these destinations depending on business-service requirements.

## Business Services and Proxy Services

Oracle Service Bus routes message between business services (such as enterprise services and databases) and service clients (such as presentation applications or other business services) through proxy services. The following sections detail the Oracle Service Bus features available for designing and implementing proxy services that support content-based routing.

### Proxy Services

Proxy services are definitions of generic intermediary Web services, that are hosted locally on Oracle Service Bus. A proxy service communicates with other services in the IT infrastructure through interfaces, which may or may not be identical to that of a service provider or service consumer business service. Proxy services can route messages to multiple business services, using their configured independent interfaces. For more information on proxy services, see [Proxy Services](#) in *Using the Oracle Service Bus Console*.

Proxy services can be defined and configured using the Oracle Service Bus Console. They are configured by specifying its interface, type of transport it uses, and its associated message processing logic. Message handling capabilities of a proxy service are implemented with message flow definitions. For more information on message flow definitions, see [Modeling Message Flow in Oracle Service Bus](#) in *Oracle Service Bus User Guide*.

When a proxy service interfaces with multiple business services, a message flow definition is configured to route a message to the appropriate business service and map the message data into the format required by the business service's interface. For more information about the structure of proxy services, see [“Message Flow Modeling”](#).



## Business Services

Business services are Oracle Service Bus definitions of the enterprise services that exchange messages during business processes. A business service and its interface can be defined and configured using the Oracle Service Bus Console. A business service is configured by specifying its interface, type of transport it uses, its security requirements, and other characteristics. A business service definition is similar to that of a proxy service, but it does not have a pipeline. For information on how to configure a business service using the Oracle Service Bus Console, see “Adding a Business Service” in [Business Services](#) in the *Using the Oracle Service Bus Console*.

A *service account* can be created to provide authentication when connecting to a business service. It acts as an alias resource for the required username and password pair. For more information, see [Business Services](#) and [Service Accounts](#) in *Using the Oracle Service Bus Console*. WebLogic Server can be used to directly manage security credentials for a business service requiring credential-level validation. For more information on business service security considerations, see the [Oracle Service Bus Security Guide](#).

## Proxy Templates

Oracle Service Bus provides the ability to create a generic proxy service that accepts any SOAP or XML messages. This helps mask the underlying complexity of protocol specifications from the service consumer. The proxy service can be configured to analyze SOAP or XML messages that it receives and dynamically route the message using content-based routing logic. Generating proxy services from a generic template, also promotes dynamic protocol switching, which allows end-point protocol selection to be made at run-time.

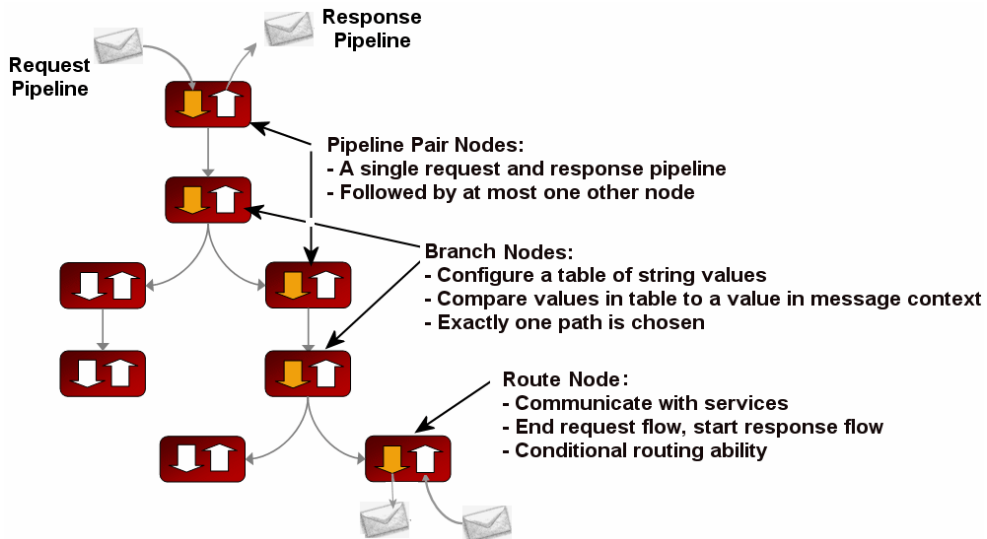
# Message Flow Modeling

Message flows are definitions used for implementing proxy services within Oracle Service Bus. Message flow modeling involves configuration of message processing logic in proxy service message flow definitions. This logic includes such activities as transformation, publishing, reporting and exception management. Each of these activities are configured as individual actions within the message flow. The graphical modeling tools available in Workshop for WebLogic and in the Oracle Service Bus Console can be used to perform message modeling.

Oracle Service Bus proxy service implementations are defined in message flow definitions using components such as pipelines, branch nodes, and route nodes. The following figure shows a high-level view of the message flow definition components.

For more information on message flow modeling, see [Modeling Message Flow in Oracle Service Bus](#) in the *Oracle Service Bus User Guide*.

Figure 5-1 Message Flow Components



## Message Pipelines

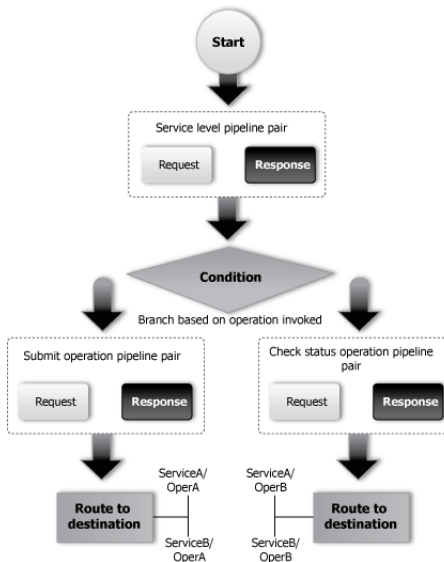
A pipeline is a named sequence of stages, representing a non-branching one-way processing path. It is used to specify the message flow for service requests and responses. Pipelines fall into one of the following three categories:

- **Request Pipelines:** used for processing the request path of the message flow
- **Response Pipelines:** used for processing the response path of the message flow
- **Error Pipelines:** used as error handlers

## Operational Pipelines

A single service level request pipeline in a stage might optionally branch out into operational pipelines (at most one per operation, and optionally a default operational pipeline). The operation is determined by user-selected criteria. The response processing starts with the relevant operation pipeline which then merges into a single service-level response pipeline. The following figure shows an example of operation pipelines in a proxy service.

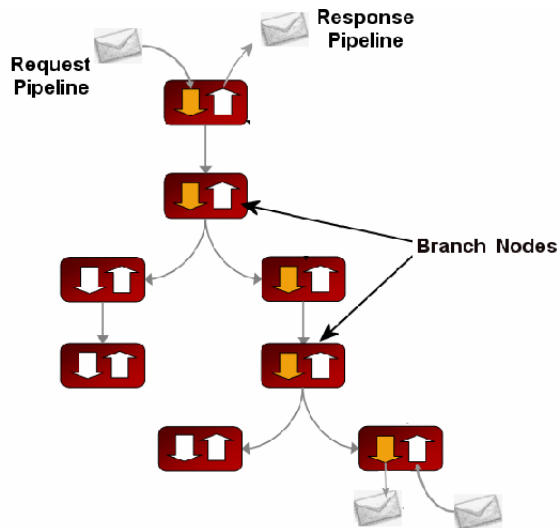
Figure 5-2 Sample Operational Pipeline



For one-way operations, the response pipeline is executed with an empty message. This permits a response to be constructed for the proxy service, enabling bridging between request/response and one-way operations. The bridging mechanism means that proxy service input can be one-way while its output is request/response or *vice versa*. The proxy service either absorbs the response from the invoked service or generates one for the client. Actions in the response flow may also be used to do post processing on the message after it has been routed to the business service or the proxy service.

## Branch Nodes

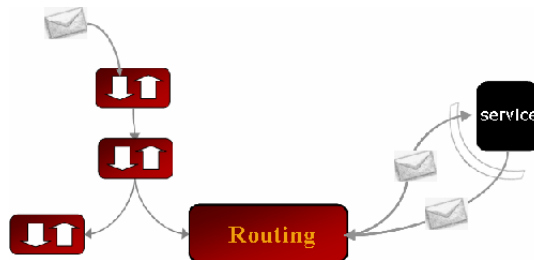
A branch node allows processing to proceed down exactly one of several possible paths. Branching is driven by a simple lookup table with each branch tagged with a simple but unique string value. A variable in the message context is designated as the lookup variable for that node, and its value is used to determine which branch to follow. If no branch matches the value of the lookup variable, then a default branch is followed. The value of the lookup variable must be set before reaching the branch node. This approach ensures that exceptions do not occur within the branch node itself. A branch node may have several descendants in the message flow tree: one for each branch including the default branch.

**Figure 5-3 Branch Nodes in a Message Flow**

## Route Nodes

The route node is used to perform request and response communication with another service. It represents the boundary between request and response processing for the proxy service, and therefore, cannot have any descendants in the message flow tree. When the route node dispatches a request message, request processing is considered finished. When the route node receives a response message, response processing begins.

The route node is very flexible in its specification and supports conditional routing as well as outbound and response transformations. It allows `if` structures and `case` structures to be combined (and nested) to define a single endpoint and operation to route the message. For information about how to configure route nodes, see “Adding a Route Node” in [Proxy Services](#) in *Using the Oracle Service Bus Console*.

**Figure 5-4 Proxy Service Route Node Communicates With Services**

Echo node is a route node that routes (or echoes) a message from the end of the request pipeline to the start of the response pipeline. The message is not routed from the proxy service to another service, but remains within the proxy service.

## Pipeline Pairs

Pipeline logic occurs in pairs of definitions consisting of a *request pipeline* definition and a *response pipeline* definition. The request pipeline definition specifies the actions that Oracle Service Bus performs on request messages to the proxy service before invoking a business service or another proxy service. The response pipeline definition specifies the processing that Oracle Service Bus performs on responses from the service invoked by the proxy service before the proxy service returns a response. Routing is performed by a route node at the end of the message flow.

To create the request and response paths, request and response pipelines are paired together and organized into a single-rooted tree structure. A branch node allows these pipeline pairs to be executed conditionally, and route nodes at the ends of the branches perform the request and response dispatching. A pipeline tree chains together instances of the following top-level components - pipeline pair node, branch node, route node or echo node.

A pipeline pair node ties together a single request and a single response pipeline into one top-level element. Only the request pipeline is executed during request processing, and only the response pipeline is executed when reversing the path for response processing.

## Pipeline Execution Stages and Actions

Each pipeline is a sequence of stages that contain actions. An action is a user-configured processing step such as transformation or publishing. Messages fed into the message flow are accompanied by a set of *message context variables* that contain the message contents and can be accessed or modified by actions in the pipeline stages.

The following table describes the actions supported in a Oracle Service Bus pipeline stages, branch and route nodes.

**Table 5-1 Oracle Service Bus Actions**

Action <sup>1</sup>	Summary Description
Communication	
Dynamic Publish	Publish a message to a service identified by an XQuery expression
Publish	Publish a message to a statically specified service.
Publish Table	Publish a message to zero or more statically specified services. Switch-style condition logic is used to determine at runtime which services will be used for the publish.
Routing Options	Modify any or all of the following properties in the outbound request: URI, Quality of Service, Mode, Retry parameters.
Service Callout	Configure a synchronous (blocking) callout to an Oracle Service Bus-registered proxy or business service.
Transport Headers	Set the transport header values in messages.
Flow Control	
For Each	Iterate over a sequence of values and execute a block of actions.
If... Then...	Perform an action or set of actions conditionally, based on the Boolean result of an XQuery expression
Raise Error	Raise an exception with a specified error code and description
Reply	Specify that an immediate reply is sent to the invoker; can be a reply with success or failure
Resume	Resume message flow after an error is handled by an error handler.
Skip	Specify that at run time, the execution of the current stage is skipped and the processing proceeds to the next stage in the message flow.
Message Processing	
Assign	Assign the result of an XQuery expression to a context variable
Delete	Delete a context variable or a set of nodes specified by an XPath expression

**Table 5-1 Oracle Service Bus Actions**

Insert	Insert the result of an XQuery expression at an identified place relative to nodes selected by an XPath expression
Java Callout	Invoke a Java method from within the pipeline.
MFL Transform	Convert non-XML to XML or XML to non-XML in the pipeline.
Rename	Rename elements selected by an XPath expression without modifying the contents of the element
Replace	Replace a node or the contents of a node specified by an XPath expression
Validate	Validate elements selected by an XPath expression against an XML schema element or a WSDL resource
Reporting	
Alert	Send an alert notification based on pipeline message context.
Log	Construct a message to be logged
Report	Enable message reporting for a proxy service

1. For information about actions, including how to configure them, see [Proxy Services: Actions](#) in *Using the Oracle Service Bus Console*.

## Operational Branching

Since message flow is typically used with WSDL-based services, operation-specific processing must frequently be performed. Rather than requiring manual configurations of a branching node based on operation, the Oracle Service Bus provides a zero-configuration branching node that branches automatically. In other words, if no operational branching is configured for a service end point, message processing will automatically branch to the appropriate operation based on the operation specified in the message context.

## Service Callouts

Oracle Service Bus provides a service callout action that offers greater flexibility for more sophisticated message flows. Service Callouts are message processing request actions from one message flow, that invoke other services registered within Oracle Service Bus. This action is generally used in response to decisions made in complex dynamic-routing processing, or to perform message enrichment. The service callout action is used inside a message flow routing

stage, to call on the destination service to perform some action on the message. The destination service returns a response to the message flow, which gets assigned to a local variable. The variable may be used within the current message flow for conditional branching.

For information about the Service Callout functionality, see "Constructing Service Callout Messages" in [Modeling Message Flow in Oracle Service Bus](#) in the *Oracle Service Bus User Guide*.

Service callouts allow custom Java code to be invoked from within proxy services. Oracle Service Bus supports a Java exit mechanism via a Java Callout action that allows call out to a Plain Old Java Object (POJO). Static methods can be accessed from any POJO. The POJO and its parameters are visible in the Oracle Service Bus Console at design time; the parameters can be mapped to message context variables. For information about configuring a Java Callout to a POJO, see [Adding Java Callout Action](#) in *Using the Oracle Service Bus Console*.

## Transformations

Transformations are used when disparate message data types exist between source and destination services, requiring data mapping to ensure service compatibility. Oracle Service Bus supports data mapping using XQuery and the eXtensible Stylesheet Language Transformation (XSLT) standard. Messages can be transformed in two ways:

- Using XQuery or XSLT to reformat the message structure
- Manipulating message content by adding, removing, or replacing certain data elements

Transformations can be created by a developer and imported into Oracle Service Bus, or scripted using XQuery in the Oracle Service Bus Console. Transformations can occur at different locations depending on the message-flow configuration of the proxy service.

In Oracle Service Bus, the Message Flow defines the implementation of a proxy service. You configure Oracle Service Bus proxy services in the Oracle Service Bus Console, which is described in *Using the Oracle Service Bus Console*.

Oracle Service Bus supports data mapping that uses XQuery and the eXtensible Stylesheet Language Transformation (XSLT) standards. XSLT maps describe XML-to-XML mappings, whereas XQuery maps can describe XML-to-XML, XML to non-XML, and non-XML to XML mappings. For more information on transformations, see [Performing Transformations](#) in *Oracle Service Bus User Guide*.



For more information, see [XQuery Transformations](#) and [XSL Transformations](#) in *Using the Oracle Service Bus Console*. For information on using the Oracle XQuery Mapper to create XQueries, see [Transforming Data Using the XQuery Mapper](#).

## Transformation Maps

Transformation maps describe the mapping between two incompatible data types. Oracle Service Bus supports data mapping using either XQuery or the eXtensible Stylesheet Language Transformation (XSLT) standard. XSLT maps describe XML-to-XML mappings, whereas XQuery maps can describe XML-to-XML, XML to non-XML, and non-XML to XML mappings. In addition, MFL described data is automatically converted to the equivalent XML for transformation with XQuery or XSLT. The resulting XML is automatically converted to MFL if the target service requires it. For more information, see [XQuery Transformations](#) and [XSL Transformations](#) in *Using the Oracle Service Bus Console*. For information on using the Oracle XQuery Mapper to create XQueries, see [Transforming Data Using the XQuery Mapper](#).

## Message Manipulation

Message manipulation is a kind of transformation in which the contents of a message, rather than its whole structure, are manipulated to make the message compatible with the destination service. This is performed by adding, replacing or removing actions to the request or response pipelines of the message flow. The different actions available to transform a message through content manipulation are described in the following table:

**Table 5-2 Message Manipulation Actions**

Action	Description
Insert	Inserts a data element into the message. This insertion may occur anywhere within the message context as specified during configuration.
Delete	Deletes a data element in the message. This is used when the target destination does not expect a specific data element in the message.
Replace	Replaces a series of text within the message with a different series. This can be used, for example, to replace the namespace of a message.

## Error Handling

Oracle Service Bus provides robust and flexible error handling for configured services. It can handle errors in the following ways:

- Testing whether an assertion is true and sending a reply with failure in the request or response pipeline.
- Configuring the service to catch and handle the error at multiple levels including the stage, route node, pipeline, or service levels. The level configured to catch the error depends on the service behavior desired.
- Letting the default system error handler handle the error.

## Message Validation

Oracle Service Bus provides the capability for incoming or outgoing messages to be validated against a WSDL or XML schema with a validation action. This action can occur at any time within the message flow and ensures that the incoming or outgoing message is in the format expected by the destination service's consumer or provider. Messages that fail validation can log the failure or create an error. In the latter case, an error stage can be used to apply alternative actions.

Message validation can be used for service versioning to validate messages against different versions of a schema or WSDL. This is to ensure the message is routed to the proper version of the service end point, or to check whether transformation must be applied prior to sending the message.

## Error Handling Pipeline

Oracle Service Bus provides a mechanism to handle errors by allowing error handlers to be defined. An error handler is a pipeline that allows various actions such as logging, transformation, and publishing to be performed to handle errors appropriately. If an error occurs within a stage a sequence of steps are executed. This sequence of steps constitutes an error pipeline for that stage.

The error pipeline allows you to handle the error in the following ways:

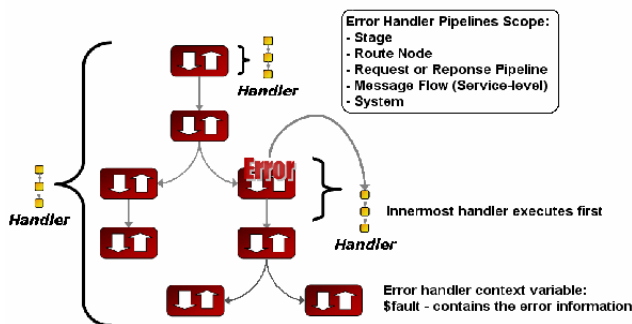
- Publish the original message to an alternate endpoint
- Formulate an error response message to be returned to the invoker of the proxy service
- Log the message

- Continue processing the message through the pipeline after modifying the context
- Raise an exception. Raising an exception transfers control to the next higher scoped error pipeline.

Errors can occur during message flow processing for various reasons. For example, security errors occur if a username is not correctly validated or authorized; transformation errors occur if Oracle Service Bus is unable to successfully transform or validate a message; a routing error is raised if a routing service is unavailable, and so on. Typically, these errors originate from a specific stage, route node or from the proxy service, as this is where most of the message flow logic is implemented.

Each stage can have a sequence of steps to execute if an error occurs in that stage. This sequence of steps constitute an error pipeline for that stage. In addition, an error pipeline can be defined for a pipeline (request or response) or for an entire proxy service. The lowest scoped error pipeline that exists is invoked on an error.

**Figure 5-5 Stage, Node, and Service-Level Error Handlers**



The Oracle Service Bus Console can be used to track messages to obtain an accurate picture of a message flow. This could enable error visibility; for example, the original reported message could be viewed, indicating it was submitted for processing, and then the subsequent reported error could be viewed, indicating that the message was not processed correctly. This would provide a complete view of both the message flow and error flow.

## Related Topics

- [Modeling Message Flow in Oracle Service Bus](#) in *Oracle Service Bus User Guide*
- [Transforming Data Using the XQuery Mapper](#)
- [XQuery Transformations](#) and [XSL Transformations](#) in *Using the Oracle Service Bus Console*
- [Adding Java Callout Actions](#) in *Using the Oracle Service Bus Console*

# Service Management

This section discusses Oracle Service Bus service monitoring and management capabilities. It is intended for system administrators and operators who manage and monitor Oracle Service Bus. It includes the following topics:

- [“Service Monitoring” on page 6-1](#)
- [“Message Reporting” on page 6-5](#)

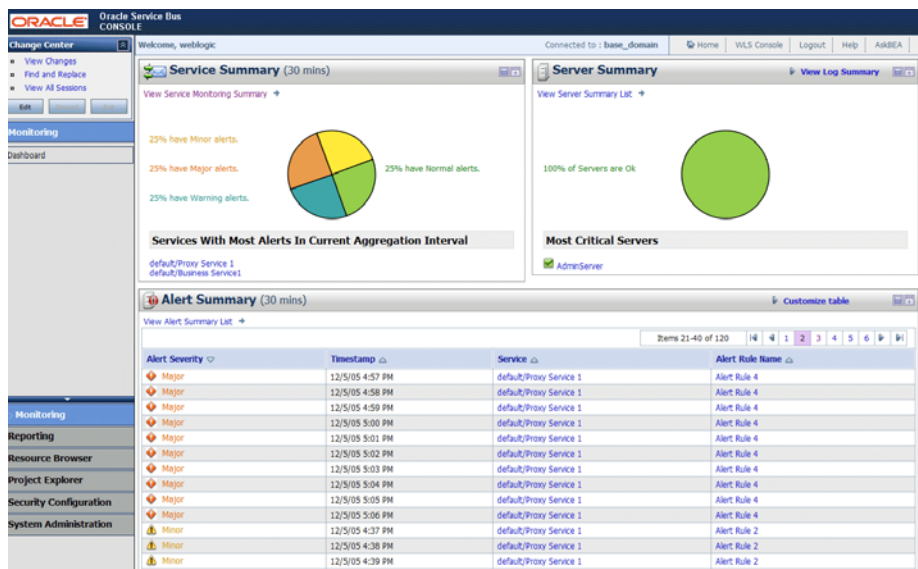
## Service Monitoring

In addition to delivering enterprise service bus capabilities such as service routing and transformation, the Oracle Service Bus also contains service monitoring and management capabilities to ensure the successful operations the IT organization expects. The following topics describe the service management and monitoring capabilities of Oracle Service Bus.

### Dashboard

Oracle Service Bus aggregates run-time statistics and allows them to be viewed in real-time on a customizable dashboard, to monitor system operational health and flag problems in messaging services, allowing quick isolation and diagnosis of problems as they occur. Oracle Service Bus Console can be used to establish service level agreements (SLAs) for the performance of a system, and configure rules that trigger alerts to provide automated responses to SLA violations.

Figure 6-1 Oracle Service Bus Dashboard



Information about system operational health can be organized by server and services. It can display status of the overall domain or the status of individual servers within it, using color-coded pie charts. It can also show service summaries showing the number of alerts and the corresponding severity for all services that have alerts defined and monitoring enabled.

In addition to the Dashboard, Oracle Service Bus provides the capability to review operational and performance statistics at individual service levels. These can be statistics for the individual service across the domain or for a specified server. It also provides performance statistics for the service at an operation level for more granular analysis.

For more information about the Oracle Service Bus Console Dashboard, see [Monitoring](#) in *Using the Oracle Service Bus Console*.

## Metric Aggregation

The information displayed on the Dashboard is based on an asynchronous aggregation of data collected during system operation. In an Oracle Service Bus production cluster domain, the Oracle Service Bus data aggregator runs as a singleton service on one of the managed servers in the cluster. Server-specific data aggregation is performed on each of the managed servers in the domain. The aggregator is responsible for the collection and aggregation of data from all the managed servers at regular, configurable intervals.

The following table lists the metrics that the Dashboard displays for each service.

**Table 6-1 Oracle Service Bus Service Metrics**

Metric	Description
Average Execution Time	<p>For a proxy service, the average of the time interval measured between receiving the message at the transport and either handling the exception or sending the response.</p> <p>For a business service, the average of the time interval measured between sending the message in the outbound transport and receiving an exception or a response.</p>
Total Number of Messages	Number of messages sent to the service. In the case of JMS proxy services, if the transaction aborts due to an exception and places the message back in the queue so it is not lost, each retry dequeue is counted as a separate message. In the case of outbound transactions, each retry or failover is likewise counted as a separate message.
Messages With Errors	<p>Number of messages with error responses.</p> <p>For a proxy service, it is the number of messages that resulted in an exit with the system error handler or an exit with a reply failure action. If the error is handled in the service itself with a reply with success or a resume action, it is not an error.</p> <p>For a business service, it is the number of messages that resulted in a transport error or a timeout. Retries and failovers are treated as separate messages.</p>
Success/Failure Ratio	$(\text{Total Number of Messages} - \text{Number of Messages with Errors}) / \text{Messages with Errors}$
Security	Number of messages with WS-Security errors. This metric is calculated for both proxy services and business services.
Validation	Number of validation actions in the flow that failed. This metric only applies to proxy services.

These metrics are aggregated across the cluster for the configured aggregation interval. The Dashboard displays information about the overall health of the system, refreshing the display at a specified interval.

## SLA Enforcement via Alerts

Oracle Service Bus provides the ability to set service level agreements (SLAs) on business and proxy services. These SLAs define the precise level and quality of service expected from business and proxy services. Rules can be configured to trigger alerts based on what the SLA measures. Multiple levels of severity can be configured for an alert including normal, warning, minor, major, critical, and fatal. Multiple alert conditions can be combined for each business or proxy service. Each alert can be based on the following parameters:

- Success rate, success ratio, failure ratio
- Message count
- Error count
- Failover/retry count
- Validation error count
- WSS error count
- Response time, minimum response time, maximum response time.

SLA alerts are set to inform the operations team of issues relating to the health of business and proxy services, or to the quality of service provided.

Oracle Service Bus implements Service Level Agreements (SLAs) and automated responses to SLA violations using *rules* that specify unacceptable service performance and the system response required under those circumstances. Rules are defined and constructed using the Oracle Service Bus Console. Oracle Service Bus evaluates rules against its aggregated metrics each time it updates that data.

When a rule evaluates to `True`, it raises an *alert*. In addition to displaying information about the alert in the Oracle Service Bus Console Dashboard, Oracle Service Bus executes the action specified for the rule when it evaluates to `True`. Any of the following types of actions can be assigned to a rule:

- Send E-mail notification
- Send a JMS message
- Send alert to the WebLogic Server Logger



It is also possible to configure operating times for alerts. Rule and alert processing is handled by the Oracle Service Bus Alert Manager. The Alert Manager resides on the same single managed server as the metric aggregator for the system.

In addition to SLA alerts, Oracle Service Bus also allows Alert actions to be configured within the message flow (pipeline alerts). These pipeline alert actions generates alerts based on message context in a pipeline, to send to an alert destination. Alert actions can be configured to include an alert name, description (which can include message elements such as \$order), alert destination, or alert severity.

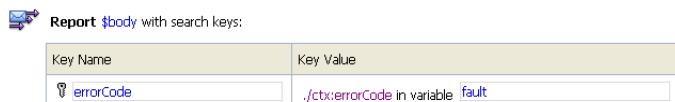
For information on how to configure Oracle Service Bus alerts, see [Monitoring](#) in *Using the Oracle Service Bus Console*.

## Message Reporting

Oracle Service Bus can report on message data as messages pass through a proxy service. This is done via a reporting action which can be placed at any point within a request/response pipeline or error pipeline stage. Reporting actions can be used to filter message information as it flows through the proxy. The data that is captured via the report action, can then be accessed via a reporting provider. The reporting actions can help determine whether there is a problem with a message pre- or post-transformation, during routing, etc.

In the reporting action, it is possible to specify information about each message that needs to be written to the Oracle Service Bus Reporting Data Stream. The following figure shows an example Report action:

**Figure 6-2 Example Report Action**



Report \$body with search keys:

Key Name	Key Value
errorCode	./ctx:errorCode in variable fault

Oracle Service Bus is packaged with a built-in JMS Reporting Provider. It picks up reported data and stores it in a message reporting database that acts as the Reporting Data Store. Customers may also Oracle Service Bus also provides a Java API for customers who wish to use their own reporting provider.

The Oracle Service Bus Console Message Reporting module displays information from the Reporting Data Store, including summary information. Message Reporting enables you to drill down from summary information to view detailed information about specific messages.

Figure 6-3 Example Message Report Summary in the Oracle Service Bus Dashboard

Message Report Summary <span>Filter</span>			
Report Index <span>▲</span>	DB TimeStamp <span>▲</span>	Inbound Service <span>▼</span>	Error Code <span>▲</span>
errorCode=BEA-382505	10/26/05 10:45 AM	MortgageBroker/ProxyServices/loanGateway3	BEA-382505
errorCode=BEA-382505	10/26/05 10:45 AM	MortgageBroker/ProxyServices/loanGateway3	BEA-382505

It is possible to customize displayed Message Reporting information by filtering and sorting the data to meet specific reporting requirements. For information on how to configure reporting actions, see “Adding an Action” in [Proxy Services](#) in the *Using the Oracle Service Bus Console*.

**Note:** Message Reporting displays information only for messages that traverse a pipeline that includes a reporting action.

Oracle Service Bus Console provides purge functionality to help manage message data. For other data management functions, standard database administration practices can be applied to the database hosting the Reporting Data Store. For a list of supported database platforms for the Reporting Data Store, see [Supported Database Configurations](#) in *Supported Configurations for Oracle Service Bus*.

Using monitoring, SLA alerts, and reporting features of Oracle Service Bus, IT operations departments can manage the health and availability of their service infrastructure in real time, measure SLA compliance, and report efficiently and effectively to their management teams and business executives.