

Application Integration Architecture Foundation Pack on Oracle Service Bus

*An Oracle White Paper
December 2008*

AIA Foundation Pack on Oracle Service Bus

Introduction	3
Oracle Application Integration Architecture (AIA)	3
AIA FOUNDATION PACK	4
Oracle Service Bus	4
AIA on Oracle Service Bus	5
Creating an Enterprise Business Service in Oracle Service Bus	6
Steps to Create an EBS on Oracle Service Bus	6
Error Handling and Test Console	12
AIA Foundation Pack Solution Recommendations	12
Conclusion	13

AIA Foundation Pack on Oracle Service Bus

INTRODUCTION

This paper discusses how AIA Foundation Pack can be used with Oracle Service Bus to provide a best of breed SOA solution that offers industry leading middleware technology with pre-built SOA integrations.

ORACLE APPLICATION INTEGRATION ARCHITECTURE

The Oracle Application Integration Architecture (AIA) heralds a new approach to integrating your application portfolio using composite business processes. It offers customers a complete, standards-based integration solution that is built for enabling IT-Business alignment and designed for extensibility and change. AIA combines the power of Oracle's Fusion Middleware along with a set of best in class application Enterprise Business Objects and Services, pre-built components and Reference Process Models for customers to create composite applications from existing investments with significantly lower cost and risk.

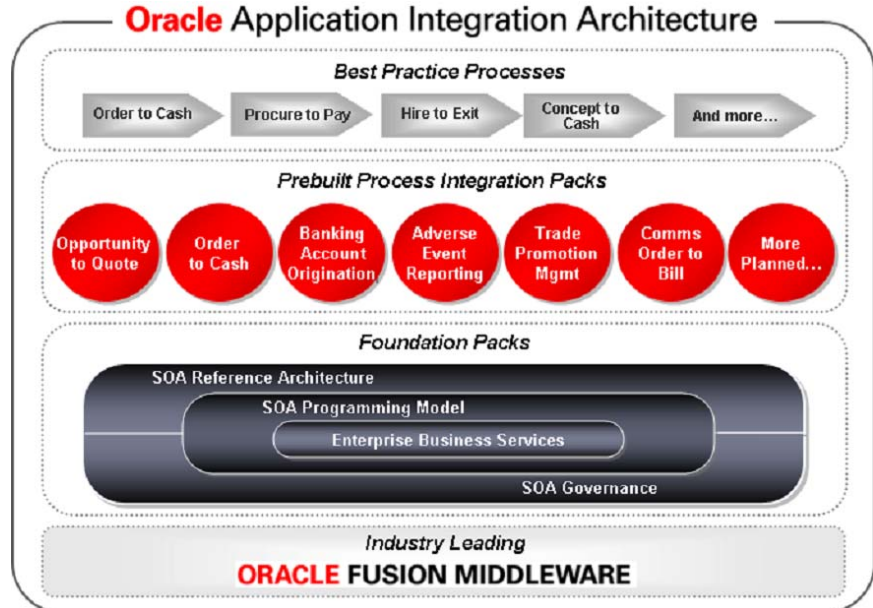


Figure 1: Oracle Applications Integration Architecture

AIA FOUNDATION PACK

Oracle AIA Foundation Pack, a core AIA product, consists of a pre-built set of Enterprise Business Objects and Services, an integration management infrastructure, and a proven methodology that significantly lowers your cost of ownership and provides a faster time to value for new composite business processes.

The Foundation Pack consists of the following components:

- Enterprise Business Objects (EBO)
- Enterprise Business Services (EBS)
- SOA Governance Tools
- Reference Architecture
- Reference Process Models
- Composite Application Framework

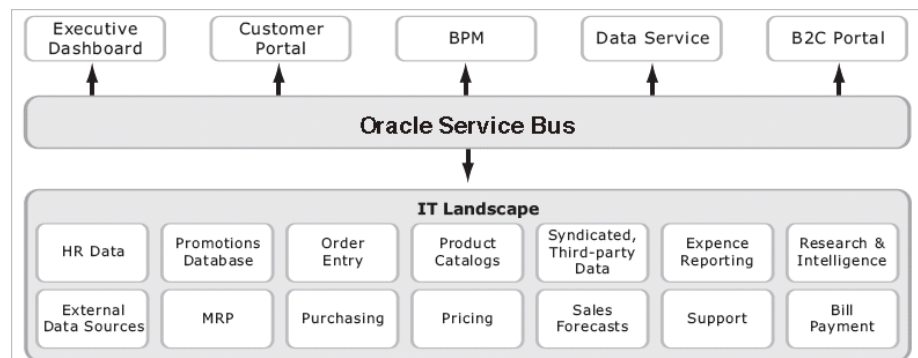
The SOA Governance suite of Foundation Pack consists of a set of tools to manage and govern your entire integration lifecycle. They include:

- Business Service Repository (BSR)
- Composite Application Validation System (CAVS)
- Composite Application Error Management and Resolution

ORACLE SERVICE BUS

Oracle Service Bus (OSB) 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.

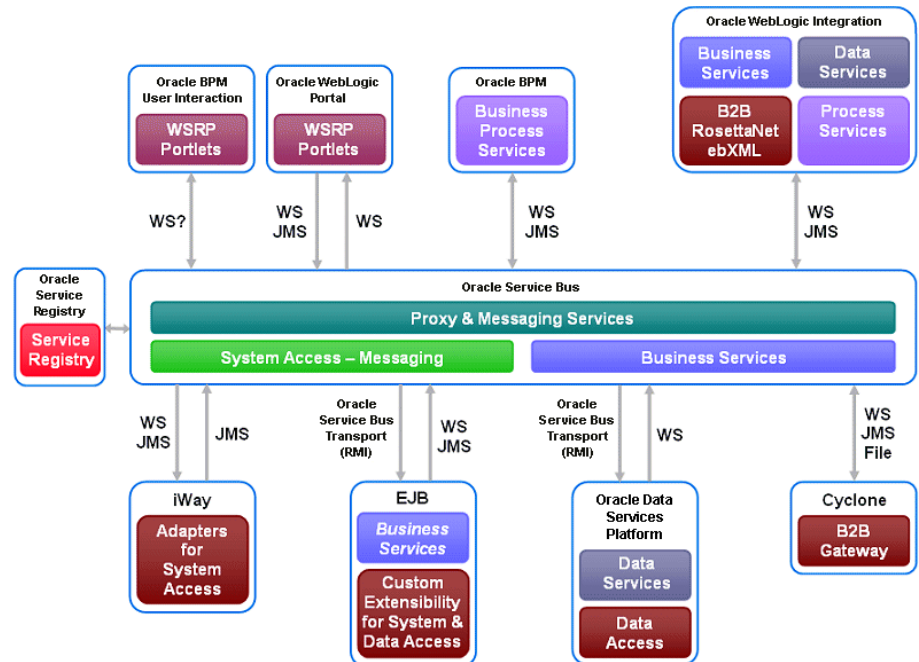
This service-infrastructure software adheres to the SOA principles of building coarse grained, loosely coupled, and standards-based services, creating a “neutral container” in which business functions may connect service consumers and back-end business services, regardless of underlying infrastructure.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



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.

AIA ON ORACLE SERVICE BUS

Oracle Service Bus has a lightweight, stateless, high-performance architecture and 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 solutions in distributed heterogeneous deployments.



Oracle Service Bus is a preferred solution for a large number of customers in various industries such as telecom, manufacturing, financial services, healthcare and public sector.

These customers are looking for newer and better ways to handle enterprise challenges such as risky, costly and delayed implementation of SOA projects. Oracle AIA along with Oracle Service Bus is well positioned to help these customers by offering best in class enterprise middleware – Oracle Service Bus with AIA’s pre-built Enterprise Business Objects, Services, integration management infrastructure and proven methodology. The Oracle Application Integration Architecture is designed to accelerate SOA implementation projects with less costs and time, and coupled with Oracle’s flagship enterprise service bus, it makes the best of breed SOA solution that is available today.

Creating an Enterprise Business Service in Oracle Service Bus

Enterprise Business Services (EBS) are the foundation blocks in the Oracle Application Integration Architecture. Enterprise Business Services represent the application independent web service definition for performing a business task. It is self-contained, that is, it can be used independent of any other services. In addition, it can also be used within another Enterprise Business Service. Enterprise Business Services are standard business level interfaces that can be implemented by the applications that want to participate in the integration.

Steps To Create an EBS on Oracle Service Bus

We recommend that an EBS be created in OSB as a proxy service. An EBS built as an OSB Proxy service inherits the intrinsic support from OSB proxy service for:

- Exposing multiple operations
- Creating Routing Rules for each operation
- Performing XSLT transformation
- Defining endpoints for each routing rule

We will describe the steps needed to create an EBS in OSB in order to illustrate how quickly and easily you can use AIA Foundation Pack and its artifacts with Oracle Service Bus. The steps are:

1. Create Project Folders
2. Upload AIA Content into OSB
3. Resolve the Schema References in the WSDL
4. Add Missing Port/Binding Information in the WSDL
5. Configure Transport for Proxy Service
6. Configure Message Flow for Proxy Service

1. Create Project Folders

In OSB environment a root project folder represents each project. The root folder for the project can be created with user defined project name. We have used following rules for creating an EBS.

- Every service will have its own project
- Each project will be represented by a root project folder
- All the artifacts of a given project would be in the same project root folder. No sub-folders are used to separate out OSB artifacts into sub-folders such as WSDL, Schemas, Proxy and Business Services.

2. Upload AIA Content into Oracle Service Bus

The AIA artifacts such as EBO and EBS are shared project resources that need to be uploaded upfront for development of any project developed over AIA programming model. These common resources can be uploaded into the OSB using OSB console by following two approaches:

- i. As a Zip File
- ii. As a Jar file

Zip File

When a zip file is used for uploading AIA content into the OSB, then step3 and step4 mentioned below need to be followed in order to resolve the schema references and provide binding information.

Jar File

However, if you decide to use a jar file, it is recommended that the schema references are resolved and the concrete port/binding information is added into the WSDLs upfront before jar file is imported into the OSB. In this case, step3 and step4 mentioned below are to be skipped.

Note: We recommend use of jar file to upload AIA artifacts into the OSB.

3. Resolve the Schema References in the WSDL

In the AIA Enterprise Services Library (ESL), WSDLs import the type definitions from separate XML schema files. When WSDLs are uploaded into OSB, the OSB cannot find the referenced schema files automatically.

OSB presents a 'conflict' message 'One of the WSDL dependencies is invalid'. Users have to manually resolve these references using the 'Edit References' screen on the OSB console, browsing for the schemas and matching them to the schemas expected by the WSDL.

4. Add Concrete Port / Binding Information in the WSDL

AIA has a generic programming model and consequently, the WSDL files in the AIA Enterprise Services Library (ESL) are abstract. However OSB requires that they also describe how to invoke the service. Therefore, WSDL files need to be updated by providing details of bindings before it can be used for creating OSB artifacts such as business service or proxy service. The following example code snippet describes typical binding information for a CustomerPartyEBS.

Example: Binding information for "CustomerPartyEBS"

```
<binding name="CustomerPartyEBSBinding"
type="corecustomerpartyEBS:CustomerPartyEBS">
  <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="QueryCustomerParty">
```

```

    <soap:operation style="document"
soapAction="SamplesQueryCustomerParty"/>
    <input>
    <soap:body use="literal"/>
    </input>
    <output>
    <soap:body use="literal"/>
    </output>
    </operation>
</binding>

```

5. Configure Transport for Proxy Service

Each proxy service is of a certain type and it can be used with the transport protocols appropriate for that type. Oracle Service Bus supports several standard and custom transport protocols. Different protocols that are available with a WSDL based services are: HTTP, JMS, Local, SB, File, FTP and WS.

We recommend use of 'local' transport for EBS proxy service. In case an EBS needs to be accessed remotely, we need to:

- Build an HTTP adapter (a proxy service using HTTP transport) for the EBS.
- Develop a business service representing this HTTP adapter proxy service for the clients who would want to access the EBS.

The end point URI is the URI, which is used by the client for accessing the service. When using local transport, the service would not have any endpoint.

The summary page of the EBS proxy service (as shown in the screen shot below) would list the chosen protocol for the service.

Create a Proxy Service - Summary (default/CustomerPartyEBS)	
General Configuration	
Service Name	CustomerPartyEBS
Description	
Service Type	Web Service - SOAP 1.1 (WSDL: EOL/AIAComponents/EnterpriseBusinessServiceLibrary/Core/EBO/CustomerParty/V2/CustomerPartyEBS, binding="CustomerPartyEBSBinding")
Transport Configuration	
Protocol	local
Get All Headers	Yes
Operation Selection Configuration	
Selection Algorithm	SOAP Body Type
Message Content Handling Configuration	
Content Streaming	Disabled
XOP/MTOM Support	Disabled
<input type="button" value=" << Prev"/> <input type="button" value=" Save"/> <input type="button" value=" Cancel"/>	

6. Configure Message Flow for Proxy Service

A proxy service's message flow definition specifies the logic that determines how messages are handled as they flow through the proxy service. A message flow definition transforms messages as needed to map the message data to the format

required by a business service (for outbound messages) or the originating client (for inbound messages). It then routes the messages to the appropriate location.

The message flow of an OSB proxy service consists of a set of nodes, of which, the 'Route' node has got a specific purpose of dispatching the message to another service. The routing behavior of the message flow is defined by the configuration of the 'Route' node.

We recommend that "Dynamic Routing" be used for configuring routing node in the message flow of an EBS.

Dynamic Routing

When configured for a route node, dynamic routing:

- Eliminates the need to hard code the routing rule expressions in the EBS configuration
- EBS does not need to be modified to change the routing endpoint
- Binding between the EBS and target service (could be either proxy or Business) is externalized out of EBS
- XQuery expression defined in a separate Xquery resource chooses one of the several services

To make the dynamic routing work, we need to have the following in place.

1. An XQuery resource with a user defined function (UDF) that

- Evaluates the routing rule's filter expression for a specific operation on the EBS.
- Specifies Routing rule's filter expression as an XPath expression
- Decides where the incoming request should be routed to
- (The UDF) Returns a complex XML node that provides fully qualified path to a Service and also the type of the service as shown below.

```
<dynamicsservice>
  <servicename>[ fully qualified path to a
Service]</servicename>
  <servicetype>[proxy/external]</servicetype>
</dynamicsservice>
```

- Every operation defined on EBS would have an XQuery resource specific to it.

- The XQuery resource for a given operation is dynamically determined at runtime.

2. An XQuery resource, XQueryResourceFinder

- For a specific operation, returns a fully qualified path to a matching XQuery resource as described above in step 1.

Note: The *XQueryResourceFinder* can be implemented in many ways. One of the options being considered is that the name of the XQuery resource to be used for a specific operation on EBS, be provided in the file *AIAConfiguration.xml*.

The value of `<servicetype>` would be either `proxy` or `external` depending on whether the EBS needs to route to a proxy or an external service (represented by a OSB Business Service).

The dynamic route is achieved by computing the value of pre-defined OSB context variable `ctx: route`, by using the values of elements `<servicename>` and `<servicetype>`. This OSB context variable is used when adding the Dynamic Route node in the message flow.

Example: XqueryResourceFinder

```
declare namespace xf =
"http://tempuri.org/XQueryResourceFinder/";

declare function xf:XQueryResourceFinder() as xs:string {
    'EvaluateRoutingRule'
};
xf:XQueryResourceFinder()
```

Example: EvaluateRoutingRule

```
(::pragmabea:global-element-parameter
parameter="$QueryCustomerPartyEBMVar"
element="v2:QueryCustomerPartyEBM" ::)
declare namespace v2 =
"http://xmlns.oracle.com/EnterpriseObjects/Core/EBO/CustomerParty/V2";
declare namespace v21 =
"http://xmlns.oracle.com/EnterpriseObjects/Core/Common/V2";
declare namespace xf =
"http://tempuri.org/EvaluateRoutingRule/";

declare function xf:
EvaluateRoutingRule($QueryCustomerPartyEBMVar as
element(v2:QueryCustomerPartyEBM))
as element(*){

if (($QueryCustomerPartyEBMVar/v21:EBMHeader/v21:MessageProcessingInstruction/v21:EnvironmentCode='PRODUCTION' or
not ($QueryCustomerPartyEBMVar/v21:EBMHeader/v21:MessageProcessingInstruction/v21:EnvironmentCode/text())) and
($QueryCustomerPartyEBMVar/v21:EBMHeader/v21:Target/v21:Applica
```

```

tionTypeCode='PORTAL' or
(not ($QueryCustomerPartyEBMVar/v21:EBMHeader/v21:Target/v21:ID/
text())))) then
  <dynamicService>

<serviceName>SamplesQueryCustomerPartyPortalProvABCSImpl/Sample
sQueryCustomerPartyPortalProvABCSImpl</serviceName>
  <servicetype>proxy</servicetype>
</dynamicService>
else
if ($QueryCustomerPartyEBMVar/v21:EBMHeader/v21:MessageProcessin
gInstruction/v21:EnvironmentCode='CAVS') then
<dynamicService>
  <serviceName>Simulator/SimulatorService</serviceName>
  <servicetype>proxy</servicetype>
</dynamicService>
  else <dynamicService></dynamicService>
};

declare variable $QueryCustomerPartyEBMVar as
element(v2:QueryCustomerPartyEBM) external;
xf: EvaluateRoutingRule($QueryCustomerPartyEBMVar)

```

Summarizing:

- We use two Xquery resources *XqueryResourceFinder* and *operation-specific XQuery resource* that evaluates the routing rule expression.
- We construct the OSB context variable ctx: route. This variable defines the dynamic endpoint of the service where message will be eventually routed.
- We use the ctx: route in the action 'Dynamic route to service' of the route node.

Modeling Route Node in Conjunction with Operational Branch

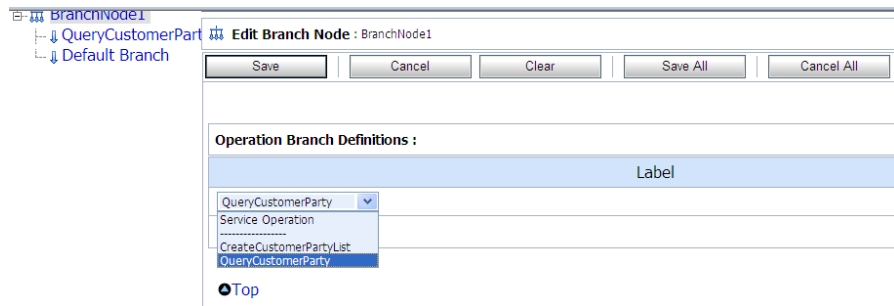
In practical situations multiple operations are defined on every EBS. Therefore it is necessary to do an operation-specific routing of the inbound message. This can be easily accomplished by using a node that automatically branches based on operations. When an operational branch node is created in a message flow then it is possible to build message routing logic based on the operations defined in the EBS.

To add an operational branch to a message flow:

- Add operational branch node
- Add operation branch definitions to it

We recommend that the dynamic routing be used in conjunction with the operational branch construct to model a routing from the EBS.

OSB Console as shown in below screen shot presents all the operations defined on EBS in the branch node configuration page, Operation Branch Definitions panel.



Note: The above steps describe how AIA Enterprise Business Services (EBS) can be built and used with Oracle Service Bus. Please note, however, that AIA Process Integration Packs (PIPs) are built using Oracle Enterprise Service Bus (OESB).

Error Handling and Test Console

AIA Foundation Pack provides features to manage errors using its Error Handling component and test your integration web services using the Composite Application Validation System (CAVS) component. However, Oracle Service Bus also has rich features to support error management and test SOA services.

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.

The recommended approach is to use Oracle Service Bus' error handling and test console.

AIA Foundation Pack Solution Recommendations

AIA Foundation Pack 2.2.1 and Oracle SOA suite 10.1.3.3 are certified on WebLogic Server 9.2 whereas the latest release of Oracle Service Bus - OSB 10gR3 runs with WebLogic Server 10.3.

We have used AIA Foundation Pack 2.2.1 with OSB 10gR3 (WebLogic Server 10.3) and recommend this combination for customers wanting to adopt the latest Oracle technologies.

We have also used AIA Foundation Pack 2.2.1 with WebLogic Server 9.2 and recommend this combination for customers who are already using Oracle WebLogic Server 9.2 and would like to continue with it.

CONCLUSION

Application Integration Architecture Foundation Pack offers a flexible, open architecture that can be used effectively with Oracle Service Bus to provide a best of breed SOA solution for customers that are looking to accelerate their SOA implementations leveraging AIA's pre-built integrations and composite business processes.



AIA Foundation Pack on Oracle Service Bus

December 2008

Author: Praveen Kumar Tiwari

Contributing Authors: Mark Craig, Ravindran Sankaran, Munazza Bukhari, Vijay Guda, Angie Wong

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

www.oracle.com

Copyright © 2003, Oracle. All rights reserved.

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

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