FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Integration using Oracle SOA Suite

Rui Pedro Silva Sá Guerra

Report of Project
Master in Informatics and Computing Engineering

Supervisor: João José da Cunha e Silva Pinto Ferreira
(Associated Professor)

3rd March, 2009

Integration using Oracle SOA Suite

Rui Pedro Silva Sá Guerra

Report of Project

Master in Informatics and Computing Engineering

Approved in oral examination by the committee:

Chair: Ana Paula Cunha da Rocha (Auxiliary Professor of FEUP)

External Examiner: José Carlos Leite Ramalho (Associated Professor of University of Minho)

Internal Examiner: João José da Cunha e Silva Pinto Ferreira (Associated Professor of FEUP)

3rd March, 2009

Abstract

This report describes an application integration effort in the context of ERPs for major corporations, using as object one of the major 3 retailers in the United Kingdom, which is intended to integrate the Oracle Retail ERP and more products from the Oracle portfolio.

Specifically, warehouse and logistics integration processes were analyzed, along with the functionalities of the system's global integration solution for Oracle Retail, the ORIB. Oracle's Service Oriented Architecture Fusion middleware application, the Oracle SOA Suite, was also analyzed.

A study of two different integration architectures was executed, ending with a description of synchronization methods that take advantage of both systems' capabilities, tested and implemented on the retailer.

Benefits and limitations of the available touch points for the participating applications are also presented, along with the complex aspects of the relational data of a retailer's transactions.

Resumo

Este projecto descreve uma integração aplicacional no sector de Enterprise Resource Planning (ERP) de retalho para grande empresas, usando como cenário um dos 3 maiores retalhistas no Reino Unido, que pretende integrar o ERP Oracle Retail e mais produtos do portfólio da Oracle.

Mais especificamente, foram analisados processos de armazéns e de logística, como também as funcionalidades da solução global de integração para a suite Oracle Retail, o ORIB. Foi ainda analisada a aplicação do porfólio Oracle Fusion middleware com arquitectura orientada a serviços, o Oracle SOA Suite.

Foi feito um estudo de duas arquitecturas de integração diferentes, terminando com uma descrição de métodos de sincronização que tiram partido das capacidades de ambos os sistemas que foram testados e implementados no retalhista.

Foram também analisados os benefícios e limitações dos pontos de contacto das aplicações em questão, bem como os aspectos complexos de dados relacionais nas transacções de um retalhista.

Acknowledgements

This project couldn't be done without the opportunity that Wipro Retail (Portugal) gave me in order to manage and implement a project on UK and I would like to thank all the support that the portuguese Wipro Retail team on the client gave me and all the Wipro Technologies co-workers that I had the change to work with.

I would specially to thank Tiago Silva and João Romualdo for all the support at management, technical and personal level they gave me within the Wipro integration team on the client we had to help.

On the other hand I would like to thank professor João José Pinto Ferreira for the support and understanding he had, knowing the technicalities of working on a project for a master, abroad and on an enterprise level.

The exhaustive work I did wouldn't be possible on such a long run without the support that Raquel and my family gave me which made all the these efforts worthwhile.

Contents

List of Figures

List of Tables

Abbreviations

ABCS	Application Business Connector Service
ABM	Application Business Message
AIA	Application Integration Architecture
BPEL	Business Process Execution Language
DM	Data Migration
EAI	Enterprise Application Integration
EAN	European Article Number
EBM	Enterprise Business Message
EBS	Enterprise Business Service
EBF	Enterprise Business Flow
EBO	Enterprise Business Objects
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
FEUP	Faculty of Engineering of the University of Porto
HIT	Heritage Integration Team
MIEIC	Master in Informatics and Computer Engineer
MOM	Message Oriented Middleware
NFR	Non Functional Requirements
OC4J	Oracle Containers for JAVA
OCC	Outer Case Code
OR	Oracle Retail Suite
ORIB	Oracle Retail Integration Bus
ORMS	Oracle Retail Merchandising System
ORWMS	Oracle Retail Warehouse Management System
OSB	Oracle Service Bus
PoC	Proof of Concept
POS	Point of Sale
RBM	Retail Business Message
SME	Subject Matter Expert
SOA	Service Oriented Architecture
UPC	Universal Product Code

XML	Extensible Markup Language
XSD	XML Schema Definition
XSLT	Extensible Stylesheet Language Transformation

Chapter 1

Introduction

This document serves to document the Master in Informatics and Computer Engineering (MIEIC) final project by the Faculty of Engineering of University of Porto (FEUP) student Rui Pedro Silva Sá Guerra on the first semester of the 2008/09 curriculum year.

This report does not contain any information that could relate it to an actual real project or a company in any manner, except for the processes executed by the author. This is the reason why some specific information was left out of this document but without compromising the value added by the same.

1.1 Scope

This is an Information Technology (IT) project, specifically on the Enterprise Application Integration (EAI) area, as part of the continuous efforts to improve the Enterprise Resource Planning (ERP) market leader applications.

The project presented in this document is part of a longer, more complex and complete program to implement Oracle's ERPs for a big European retailer, renewing all the business supporting applications.

Customizing applications and, more important, their middleware in order to add value to the clients business applications purchases has become constant and even mandatory to the ones that have an objective to have more effective business processes. Given that, this project is critical to the actual processes that it has to support.

The Oracle Retail (OR) suite standard package comes with the Oracle Retail Merchandising System (ORMS), Oracle Retail Warehouse

Management System (ORWMS) and full integration between these two applications, being the ORMS the master application for foundation data (Appendix A).

ORWMS will be the first Oracle Retail (OR) suite application to go live in the client application roadmap and it has been decided by the client, after a risk analysis, that the first warehouse where ORWMS will be implemented will be a freezer warehouse, due to the simplicity of the items foundation data compared to ambient and fresh products Mainframe, the client's main legacy application and foundation data provider, which will have to replicate the integration functionality provided in the end state by ORMS, will be required to provide foundation and transactional ordering data to ORWMS; and ORWMS will provide inventory and order fulfilment data to Mainframe processes.

This project is in the first steps of the programme's transition plan and will be discontinued as soon as all the business processes start to use Oracle Retail Merchandising System as the master application.

1.1.1 Project

ORWMS is an auxiliary system on the Oracle Retail suite, since it depends on a central merchandising system like ORMS for foundation and ordering data and is single instanced (each ORWMS installation only serves one warehouse).

On the other side, the client's central legacy application holds all the data and changes required for ORWMS to work without a full master system; therefore it's required to integrate Mainframe with ORWMS for foundation and system data.

Since the client wanted to have a full OR application working before having the risk of a blind and full OR suite installation and ORWMS gathers all the pre-requisites to have a standalone installation with minimum customization, it has been decided that ORWMS will be the first OR suite application to go live and that it will serve one or more warehouses depending on the outcome of the first implementation.

Oracle Retail Integration Bus (ORIB), which is the most complete out-of-box integration middleware of the OR suite, includes routing functionality required for a possible multi-instance installation of ORWMS, so the only business functional requirements for this project are limited to message forwarding and internal enrichment, having in mind that the Mainframe processes will deal with message transformation to cope with the ORIB payload format.

Since Mainframe data storage is file based and Oracle Fusion Middleware (OFM) connectivity does not support the Mainframe operating system (z/OS), WebSphere MQ Series, a message queue system, is the only viable option (and most reliable) to connect any Oracle application with Mainframe on this client.

The following diagram represents the project components from end to end, with constrains described and which data interfaces were required by the client to comply with their processes.

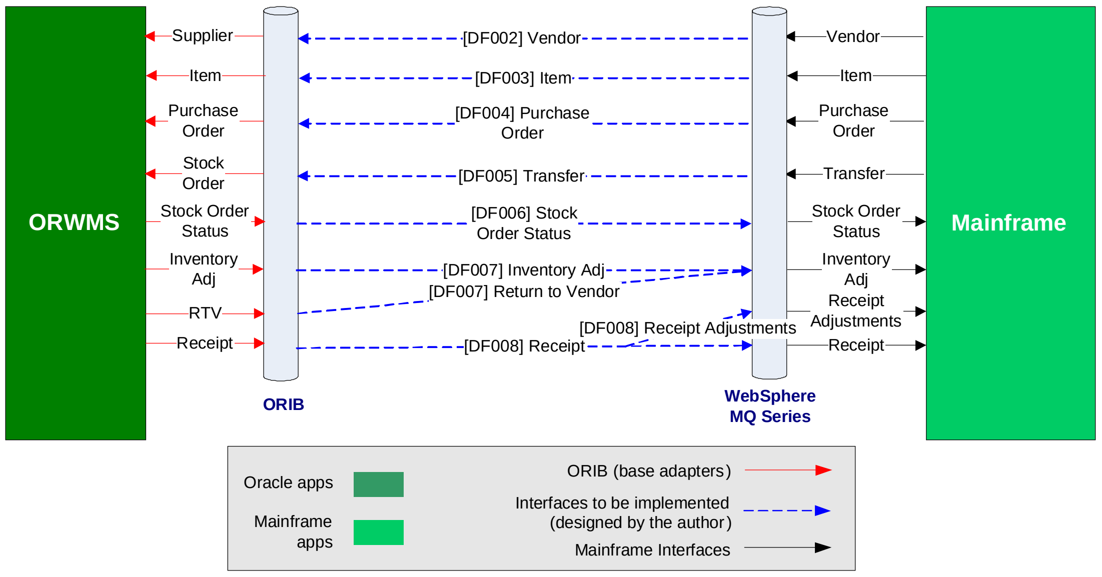


Figure 1.1: Project components overview

This project's roadmap starts with Proof of Concept (PoC) on the connectivity, analysing the several OFM solutions [OFM] (i.e. Oracle Service-Oriented Architecture suite) that the client has bought to present a solution that can cope with the functional and non functional requirements for the final integration system. The outcome expected from the PoC is a report of the verification and validation of the integration between Mainframe and ORWMS, having as constrains a fixed length element based messages (via WebSphere Message Queues Series system) on one side and the ORIB on the other side, as integration components touch points. This analysis includes the verification of non-functional requirements as message ordering and error handling, notification and recovery.

The next project step is the design phase where it's required to produce a functional design document, followed by a technical design document and ending with the actual development phase for all the components.

The project roadmap ends with the unit, performance and end to end component testing, and go-live support.

1.1.2 Responsibilities

This project was developed by the author and integrated in the Wipro Retail programme for the client.

All the project steps described are of the author's responsibilities except for the design, which is a responsibility shared between the author and the client architecture team. System testing and data migration design are responsibilities of the correspondent Wipro Technologies teams allocated for the client. The component development and component unit testing process was done by the Wipro Technologies offshore team allocated for this project.

The following table presents an overview of the involvement of the author in this project.

Table 1.1: Author's involvement on the project

Tasks	Involvement		Responsibilities
	Full	Partial	
Definition (PoC)	•		Proof of concept development, report and presentation
Integration Framework		•	Give project inputs and constraints on the programme framework for integration
Conceptual Design Analysis	•		Document owner and head designer receiving inputs from the client
Functional Design	•		Document owner and head designer receiving inputs from the client
Technical Design	•		Document owner and head designer receiving inputs from the client
Client's Legacy Design		•	Active support on legacy design helping the client understand what is expected and what to expect
Logistics and ORIB as-is		•	Logistics team support for ORIB interfaces and to be developed components
Design Review		•	Updating conceptual, functional and technical design according to client's input
Development Phase		•	Functional support, technical support and on site component testing
SIT and UAT		•	Functional and technical support
Dry-run		•	Functional and technical support
Go Live		•	Functional and technical support
Rollout Hypercare		•	Functional and technical support

1.2 Motivation and Objectives

The motivation for the author to work on this project is based on his in special interest on the EAI area and also on the logistic business processes, gathered along with his experience working face to face with clients on long projects.

Having a great middleware available for development from Oracle and a new reference architecture for it (Oracle Application Integration Architecture – AIA) also made this project a must do for him.

The objectives for this project are based on the relation between the client and the development team, working on the analysis, design and specification, along with the responsibilities of the development of the components, component testing and go live support.

It was also required to have full understanding of Oracle SOA Suite, particularly Oracle Business Process Execution Language (BPEL) implementation and Oracle AIA as an integration framework.

The actual objectives are described on the next subsections.

1.2.1 Deliverables

The following deliverables are part of this project:

- Proof of Concept Report Document;
- Requirements Definition Document;
- Functional Design Document;
- Technical Design Document
- Unit Test Plan;
- Unit Test Cases;

The following documents are artefacts for this project and were produced to support the deliverables:

- Infrastructure Deployment Profile Document;
- Failure Point Analysis;
- Fixed Length Message Definitions for the Mainframe WebSphere MQ Series touch point.

As described, the project main objective is to integrate ORWMS with Mainframe for the data interfaces mentioned, using ORIB topics and WebSphere MQ Series queues as touch points, therefore the components used are described along this document on and are the main deliverables.

1.2.2 Support

Since this project crosses several team scopes, it has became as part of the project to give support to the Mainframe Heritage Integration Team (HIT) from the client side, responsible for the design and development of the processes between the Mainframe data and the WebSphere MQ Series queues; the Wipro Technologies Data Migration (DM) team for this logistics project which is required to use the integration interfaces for the suppliers and items initial ORWMS load; the Wipro Technologies Testing team for this logistics project which is required to test both ORWMS and the integration layer that connects to it and, more closely, the Wipro Technologies Offshore team for the client's logistics integration which is responsible to develop the components specified.

1.3 Document Structure

This report has five chapters.

The second chapter describes the concepts behind the project developments. Retail Industry and IT, Enterprise Resource Planning suites, integration architectures and some technologies involved with the developed solution are some of the concepts discussed on this chapter. It also gives some background on some tools that were used during the project and a comparison between the several integration logical architectures (or patterns) that could be used.

The actual project description is presented from a problem solving point of view, having always in mind that it's a constrain not to customize the target applications and to use middleware as the develop method and base architecture to integrate them.

The introduction to the problem that is the scope of this project and the high level description of the solution itself is presented on the third chapter, along with an overview of the proposed solution to the client, considering constrains, functional and non functional requirements.

The forth chapter presents a detailed overview of the implementation of the proposed solution, enriched with post develop phases documentation (i.e. initial target application load and testing).

The final chapter presents the conclusions about this project and evaluates the given results with some highlights on the work that still needs to be done.

Chapter 2

Bibliographic Revision

This chapter describes the state of the art for this project and describes the background context behind its scope. It also presents a technology review of the major architectures, software and tools used in the context of this project.

It is structured in a top-down manner: starting with the business overview, in section 2.1, followed by the applications that support the business, with added relevance of the middleware that glues the applications (in section 2.2), to the integration architectures (in section 2.3) and ending in the technologies (in section 2.4) that are the infrastructure for the applications and relevant tools (in section 2.4) for this project, as shown in the next diagram.

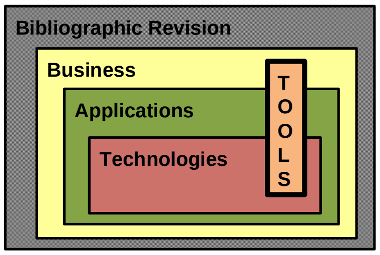


Figure 1.2: Bibliographic revision structure

2.1 Introduction to Retail

2.1.1 Retail Industry

From an enterprise process point of view, a business with big volumes of sales and a complex supply chain usually have complex, critical business processes and information flows that are unique in its sector and, in many cases, unique in each company.

The retail activity is the biggest world business area with more than 5.000.000M€ [EIM] and, on this business, most of the processes and cash flows come from different and retail specific business activities which differs from the normal companies where the core processes are usually the same.

The retail business is based on merchandising management; items from the supplier are processed along the supply chain until they get to the Point of Sale (POS), with transportation, warehouses and clients involved.

Giving special emphasis to the warehouse specific processes, which are part of the logistics and supply chain management processes of a retail company, they control the storage of the warehouses (normally finished goods), tracking, receiving, put away, what's required to return to vendor and shipping. Warehouse management systems are a must have on a retail company since they provide cutting edge value due to the simplicity of the process tracking and repetitiveness.

2.1.2 IT in Retail

The processes and automatic information support for all the business related activities on a retailer have direct impacts on the levels of productivity and, therefore, the success of the company.

In this business, it's impossible to manage items or products and track the stock status without having any technology to support the business. The profit of a retail company is directly related to its capacity to manage warehouse and store operations in real time, being the reason why they usually look for systems which can manage all these processes with on the fly High Availability.

Besides having this type of infrastructure requirement, even more system support related requirements are requested as, for example, intermediate data persistence and two-phase commit for data interchange. This is required in order to ensure that the data sent from one application to another one or more is not lost in the process or, more critical in cases of delta changes messages, information is sent more than once.

These technology requirements were already created a long time ago for other critical related systems and, since they work perfectly out-of-box on the retail industry, the actual innovation and work for IT providers on retail is in creating applications and applications customizations that adapt to the very specific retail industry business processes. These customizations can be easily proven to give a Return of Investment (ROI) in a matter of months and that’s why the retail industry in the last couple of years started thinking about contracting companies as Wipro Technologies as their systems integrator and trust them to change completely their IT systems.

2.1.3 Integration Value

An integration of ERPs or standalone business applications is usually required by the business heads of a company and requires a big effort from the operational resources in order to cope with those requirements. These integration requirements are normally based on the view that the decision-makers have of the subjacent business processes which can not represent the actual business needs. This is a situation where the entity that requires one type of behaviour does not own the responsibilities for that behaviour but other one does.

The solution to this issue is to give the power to the operational resources to define which are the integration needs and present them to the decision makers on an “integration value proposition” basis and to create value observation processes in order to define whenever they meet the strategy defined by the business leads or not [IVPP].

2.1.4 Integration in Retail

Most of the retail oriented ERPs are modulated, for instance, usually the central processes are managed in the central application and the warehouse processes in a logistics or warehouse support application with its specific functionalities for the inherited business processes. Some retailers are even advised to buy different applications from different business software companies in order to obtain the best out-of-box for each working module or group of business processes.

This is especially true valid with central and warehouse applications which explains, for example, why the Oracle Retail suite includes a complete online integration solution between ORMS and ORWMS via RIB for the vanilla installation.

When these applications come from different software companies it’s required for a system integration consulting company to step in and to develop integration processes between the two (or more, i.e. voice picking in a warehouse module) applications. Usually, the slave applications like the warehouse are prepared for this integration somehow providing the systems integrator a less intrusive and application dependent integration.

2.2 Suites / ERPs

2.2.1 Oracle Retail

The Oracle Retail Suite works with two different paradigms: J2EE and PL/SQL. The first one uses Oracle Containers for Java (OC4J), which means that Enterprise Java Beans (EJB) are hosted on a Java container and the PL/SQL paradigm deployed using an Oracle Database 10g.

The OR suite contains more than twenty applications, for example the OR Merchandising System (ORMS), OR Price Management (ORPM), OR Warehouse Management System (ORWMS) and OR Invoice Matching (ORIM), among many others. These applications were developed by the Retek software house and were attached to the Oracle portfolio when Oracle bought Retek in 2005 [OBRS].

Oracle Retail Warehouse Management System

This application (part of the Oracle Retail's Supply Chain Planning and Execution solution group) architecture is N-tier (each application’s instance serves one physical warehouse) and the Graphical User Interface (GUI) is web-based [ORWMS1].

ORWMS is the master OR application for business modules such as inbound freight scheduling, automatic shipping notice (ASN), trailer management, stock receiving, inventory control, warehouse space utilization, shipping, stock return to vendor (RTV), among others.

ORWMS uses mainly ORIB for OR applications integration but has integrated radio frequency (RF) handheld control for warehouse operational activities [ORWMS2].

Oracle Retail Merchandising System

ORMS is the central OR management system and stands on an Oracle Database 10g and the GUI is developed with Oracle Forms. Most of the OR communication processes like the daily reception of receipts or shipments are present on ORMS since the control point for these processes must be central from a retail processes point of view.

ORMS also manages another retail business processes such as purchase orders, stock orders, deals, simplified price management, replenishments, forecasting, etc [ORMS2].

Oracle Retail Integration Bus

ORIB is the principal integration application of the OR suite and it uses the Java Messaging Service (JMS) topics interface to implement the publisher / subscriber paradigm. The actual JMS implementation is, in its version 13, accomplished using Oracle Advanced Queues (AQ).

The persistent model for ORIB is based on two-phase commit XA [XAJV] which ensures the delivery of the messages to all the consumer systems via Enterprise Java Beans, the ORIB adapters.

The ORIB adapters are responsible for the message forwarding from the source applications, to topics and ending on the target applications. The publisher adapters are developed using EJBs and the subscribers are defined as a Message Driven Bean (MDB) which means that the process trigger is a message on a topic subscribed by the MDB. There is another type of adapter responsible for Transformation, Address, Filter or Route (TAFR) the messages. These TAFRs are MDBs and publish the subscribed messages on another topic if they are not filtered.

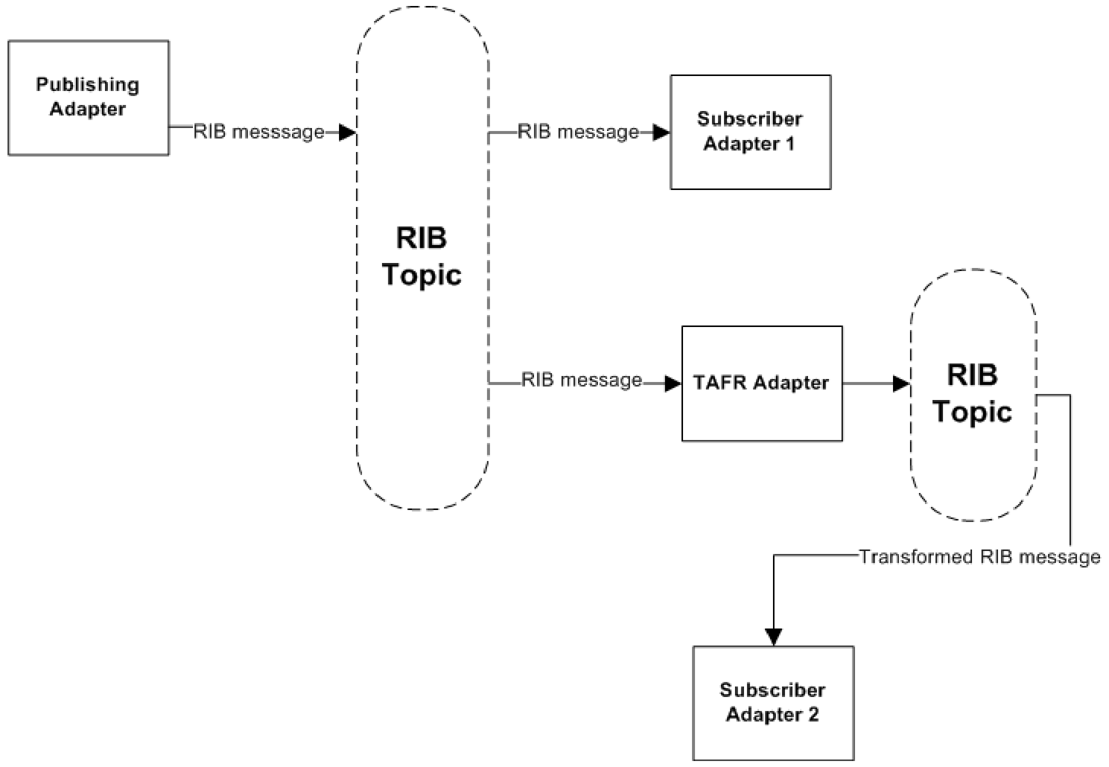


Figure 1.3: ORIB adapters [ORIBb]

On a side note, ORIB implementation can be distributed from an adapter point of view, which gives permission for a message to be send on a network using a bridge.

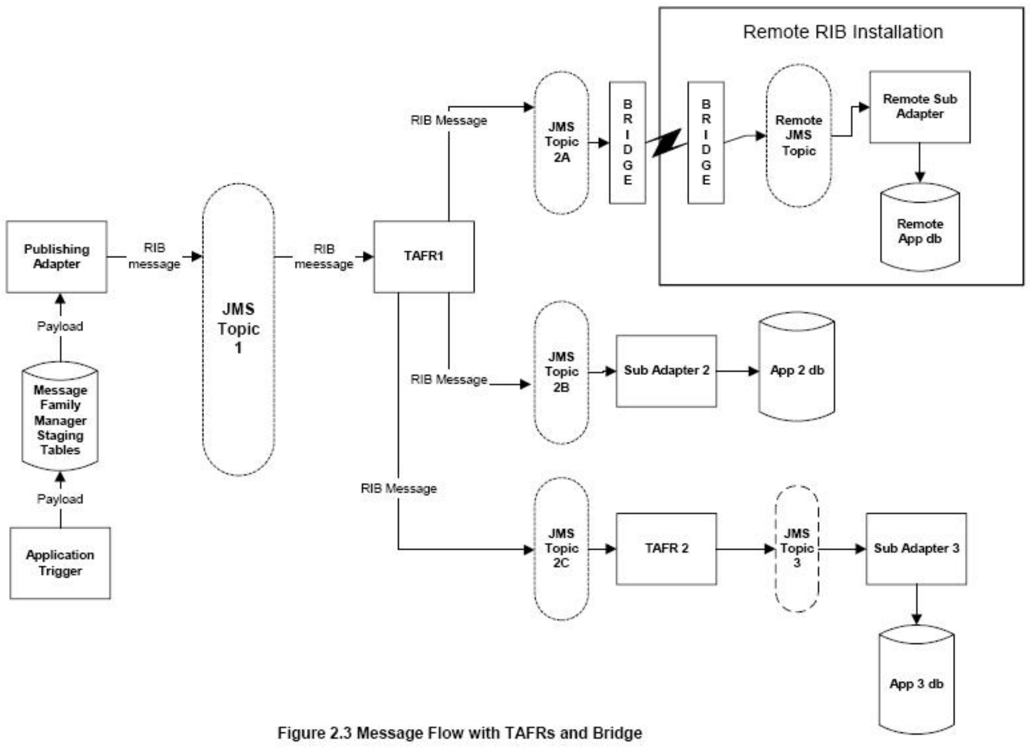


Figure 1.4: ORIB multi server installation [ORIBb]

ORIB uses an envelope XML message referred as RIBMessages which contain the information required for the adapters to perform their tasks. The actual message content is referred as a RIB payload and it is populated on the publishing application since it contains the transactional information to be transmitted. It is formatted using XML based on a XML Schema Definition (XSD).

Every RIB payload is part of a Message Family (business related, e.g. items, vendors, transfers) and is mapped to one message type (e.g. creation, deletion, detail modification).

ORIB with PL/SQL

Since ORMS and ORWMS are based on an Oracle Database, ORIB has to connect to the applications data with PL/SQL adapters.

This integration process starts with a database trigger event on the source database which fulfils a staging table with the data to be received by the ORIB publishing adapter and them the adapter, which is polling that table, starts the normal publishing process.

The RIB payloads are created by different manners, depending on the data interface, due to historical reasons of the applications: via Character Large Object Binaries (CLOBS) on the database table, which contain the message XML, or via Oracle Objects, which are quicker since are based on the database native data and Object Oriented (OO) programming concepts.

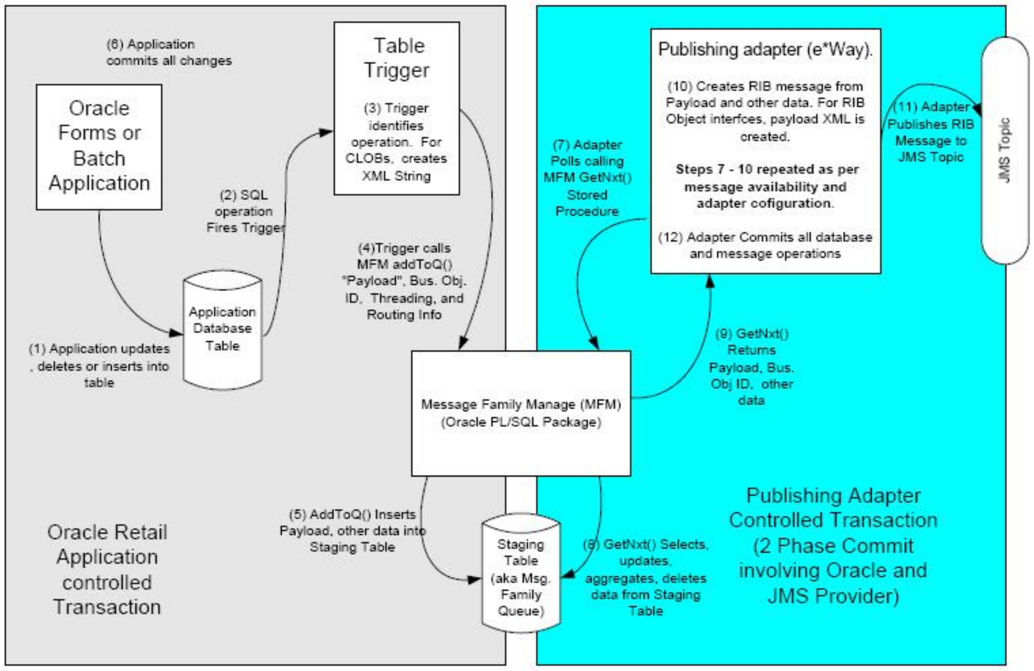

(6) Application commits all changes
Oracle Forms or Batch Application
Table Trigger
(3) Trigger identifies operation. For CLOBs, creates XML String
(2) SQL operation Fires Trigger
(4)Trigger calls MFM addToQ() "Payload", Bus. Obj. ID, Threading, and Routing Info
Application Database Table
(1) Application updates , deletes or inserts into table
Message Family Manage (MFM) (Oracle PL/SQL Package)
(5) AddToQ() Inserts Payload, other data into Staging Table
Staging Table (aka Msg. Family Queue)
Oracle Retail Application controlled Transaction
Publishing adapter (e*Way).
(10) Creates RIB message from Payload and other data. For RIB Object interfces, payload XML is created.
Steps 7 - 10 repeated as per message availability and adapter cofiguration.
(12) Adapter Commits all database and message operations
(7) Adapter Polls calling MFM GetNxt() Stored Procedure
(11) Adapter Publishes RIB Message to JMS Topic
JMS Topic
(9) GetNxt() Returns Payload, Bus. Obj ID, other data
(8) GetNxt() Selects, updates, aggregates, deletes data from Staging Table
Publishing Adapter Controlled Transaction (2 Phase Commit involving Oracle and JMS Provider)

Figure 1.5: ORIB PL/SQL Publishing [ORIBb]

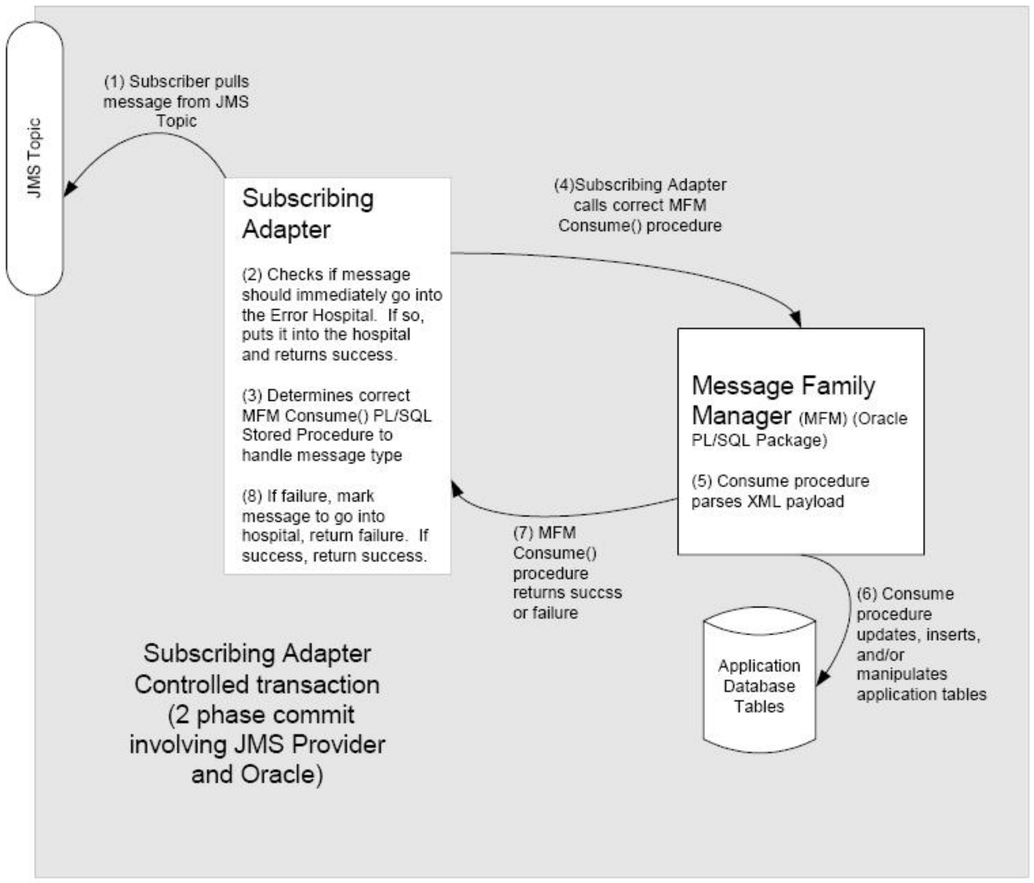

JMS Topic
(1) Subscriber pulls message from JMS Topic
Subscribing Adapter
(2) Checks if message should immediately go into the Error Hospital. If so, puts it into the hospital and returns success.
(3) Determines correct MFM Consume() PL/SQL Stored Procedure to handle message type
(8) If failure, mark message to go into hospital, return failure. If success, return success.
(4)Subscribing Adapter calls correct MFM Consume() procedure
Message Family Manager (MFM) (Oracle PL/SQL Package)
(5) Consume procedure parses XML payload
(7) MFM Consume() procedure returns succss or failure
(6) Consume procedure updates, inserts, and/or manipulates application tables
Application Database Tables
Subscribing Adapter Controlled transaction (2 phase commit involving JMS Provider and Oracle)

Figure 1.6: ORIB PL/SQL subscribing [ORIBb]

<u>ORIB Hospital</u>

ORIB includes a message logging system for the faulted messages to be repaired manually after the message publish retry count expires.

ORIB Hospital also stores the messages that follows a faulted message and are related to it. This is part of the hospital processes because, if a business entity represented by a message has faulted, the following messages referenced by the first one can depend of the correct post process of it. This functionality avoids, for example, that a vendor address modification message is processed if the vendor creation message for the same vendor has faulted.

2.2.2 Oracle SOA Suite

This Oracle suite includes a set of middleware applications and a set of service infrastructure components for creating, deploying, and managing services.

Oracle SOA Suite includes the following components:

- Integrated Service Environment (ISE) to develop services;
- Oracle BPEL Process Manager to orchestrate services into business processes;
- ESB to connect existing IT systems and business partners as a set of services;
- Oracle Business Rules for dynamic decisions at runtime that can be managed by business users or business analysts;
- Oracle Application Server Integration Business Activity Monitoring;
- Oracle Web Services Manager (for authentication, authorization, and encryption policies on services);
- Oracle Application Server 10g for a J2EE application environment. [OSSSG]

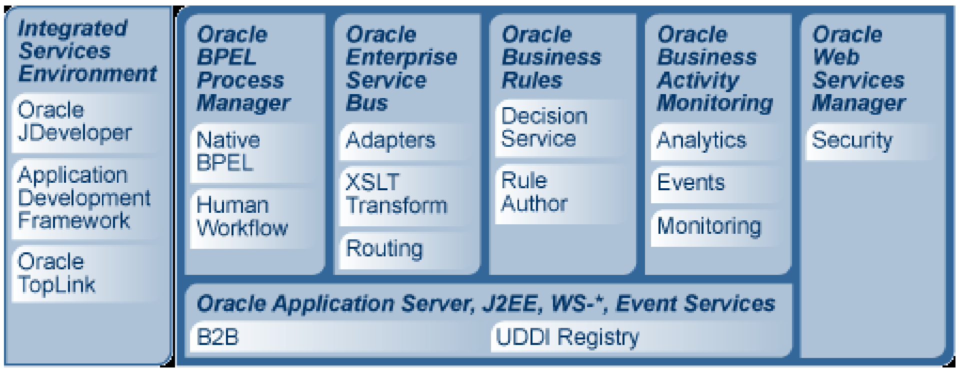

Integrated Services Environment
Oracle JDeveloper
Application Development Framework
Oracle TopLink
Oracle BPEL Process Manager
Native BPEL
Human Workflow
Oracle Enterprise Service Bus
Adapters
XSLT Transform
Routing
Oracle Business Rules
Decision Service
Rule Author
Oracle Business Activity Monitoring
Analytics
Events
Monitoring
Oracle Web Services Manager
Security
Oracle Application Server, J2EE, WS-*, Event Services
B2B
UDDI Registry

Figure 1.7: Oracle SOA Suite Architecture [OSSSG]

2.3 Online / Real Time Integration Architectures

The following chapters describe the integration methods used on this project, along with a different integration approach, the database sharing method, for comparison purposes.

2.3.1 Message-Oriented Middleware Integration

Message-Oriented Middleware (MOM) or Messaging for short is based on a loose coupled (synchronous or asynchronous) method of passing information between applications [MOMMK]. The original concept for MOM was queue based, not requiring component integration synchronization which permitted “near-real time” integration, since a queue pooling system does not permit real time message listening, due to performance issues on a large scale; but nowadays, with the help of web services and component interaction methodologies, synchronous application integration (which requires acknowledgment from the responder process) has become part of the MOM world.

From a message point of view, this integration method permits a simple First-In, First-Out (FIFO) message ordering or even message prioritization. Messages can be also defined as persistent (written to a database or a file system) and non-persistence (stored in memory) depending on the value of the message.

2.3.2 Publish/Subscribe

This integration pattern synchronizes systems with oneway propagation of messages where one or more applications publish messages into a staging area and one or more applications subscribes them [MSIPP]. There are different implementations of this pattern, for example ORIB uses the topic implementation (multi subscriber queues) where a topic serves several application publishers and subscribers, depending on the message family; on other side, WebSphere MQ Series is normally used to implement single subscriber queues, where each queue is defined to receive only one type of messages.

2.3.3 Service-Oriented Integration

“Service-Oriented Integration connects systems by enabling them to consume and provide XML-based Web services. The interfaces to these systems are described through Web Services Definition Language (WDSL) contracts. Systems interact with each other by using SOAP messages. SOAP messages are usually conveyed through HTTP by using XML serialization.”[MSIPP]

This takes advantage of the internet capabilities which is sometimes required, due to the physical location of the different applications of an enterprise. Other advantages are: open standards, application independent integration which gives system extensibility and technology independence.

The disadvantages of this integration pattern are related to the XML message parsing but, since most of the implementations of this pattern like Oracle SOA Suite already have tuned built-in XML related utilities, this disadvantage is usually disregarded.

2.3.4 Shared Database

This is a tightly coupled integration method on which the integrated applications read and write to the same physical database (or even to the same database schema). An example of this is shown on the integration between the Oracle Retail applications Merchandising System and Price Management [ORPMIG]. There are several approaches to achieve this type of integration but the most common one is to develop database packages (composed by procedures and triggers) to read and write directly to the database without requiring external components.

2.4 Technologies / Frameworks

2.4.1 Business Process Execution Layer

BPEL is an executable language for web-services interaction and requires an engine to read the actual code. Oracle BPEL Process Manager (which is part of the Oracle SOA Suite) is an orchestration engine which fully implements this language.

This language, which is based on serialized XML, objective is to support business transaction [WBPEL].

The Oracle BPEL implementation includes the SOA suite infrastructure functionalities and a console for process (developed components) management. This console provides process-level logging, error handling and notification as well as component deployment

JDeveloper, an IDE from Oracle, is the main tool for the development of BPEL processes for the Oracle BPEL process manager tough not required since the deployment can be done via a web browser on the BPEL console. The development of BPEL processes from JDeveloper includes graphical interface for BPEL activities and partner links (web-services interaction specification) as well as Oracle adapters development and deployment (with web services interface).

There are several types of BPEL activities. The most common ones are the receive activity (to receive the web service invocation value), the invoke activity (to send the value of a BPEL variable on a web service invocation), the assign activity (variable assignation with copy, append or remove) and the transform activity (an assign activity mapped into a XLT transformation file). Many more structural and action based activities are supported in BPEL.

The error handling paradigm for BPEL is based on exceptions which are dealt by catch branches that can include BPEL activities.

The following diagram is an example of a synchronous BPEL process created using JDeveloper.

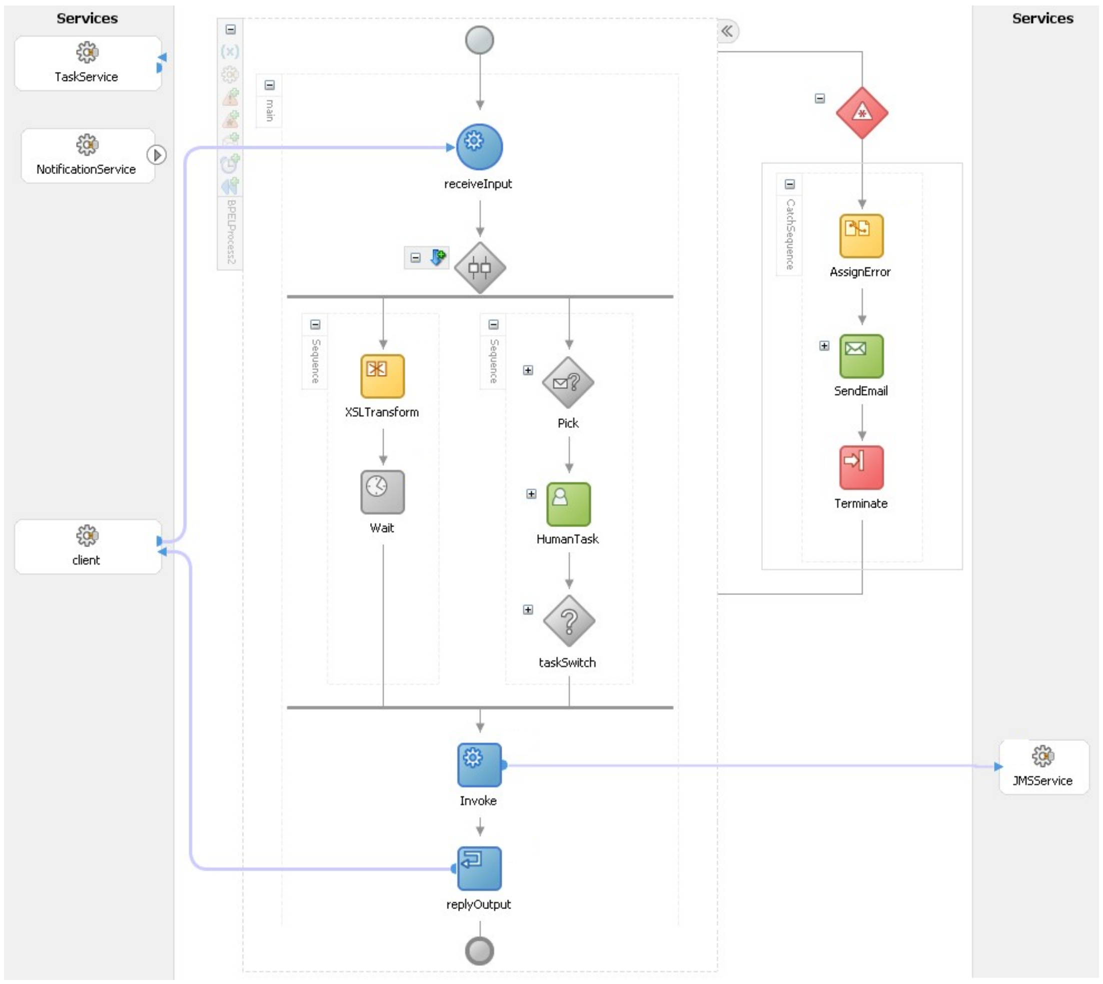


Figure 1.8: BPEL process example with JDeveloper

Oracle Application Integration Architecture (AIA) uses Oracle BPEL process manager for the Application Business Connector Services (ABCS) implementation and that implementation detailed on section 2.4.6 of this document.

2.4.2 Oracle Enterprise Service Bus

The Oracle ESB provides the following functionalities:

- Connectivity via Oracle adapters, Web Services, databases, JMS, etc;

- Message Transformation (including domain value transformation);
- Message Routing.

This Oracle fusion middleware application best intent is to provide a service virtualisation layer that decouples services from each other, whether as part of a point to point integration or as part of a BPEL process. The Oracle ESB is built on many of the same components as the Oracle's solution for BPEL. Some shared components include:

- Common adapter framework;
- Enterprise Messaging System;
- JDeveloper based designer;
- XSLT Transform Tool.

This type of integration application should be used when a very lightweight point to point integration is required, without special connectivity properties such as message headers. This application is also the most complete in terms of service virtualisation and service dependency analysis.

2.4.3 JAVA Message Service

The Message Oriented Middleware (MOM) specification from Java is the Java Message System (JMS) and defines the interfaces for producing or consuming message objects and also the objects where the messages are stored. These can be queues or topics (multi consumer queues).

A queue is a message row on which messages are consumed on a First In, First Out basis (FIFO) and are deleted upon read acknowledgement, therefore having a constraint of being a single consumer staging object.

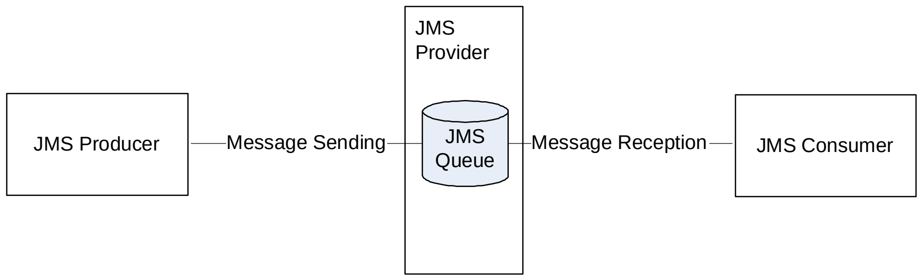


Figure 1.9: Queue system model

The JMS interface for the publishing and subscribing paradigm is considered a topic.

The concept behind topic is to permit that more than one consumer can access the same message. This can be achieved by two different methods: a persistent one, where the subscriber registers itself on the JMS for a topic and messages are only deleted from a topic when all the registered subscribers receive that message; or by a non-persistent way, where a subscriber must be active in order to receive messages from a topic.

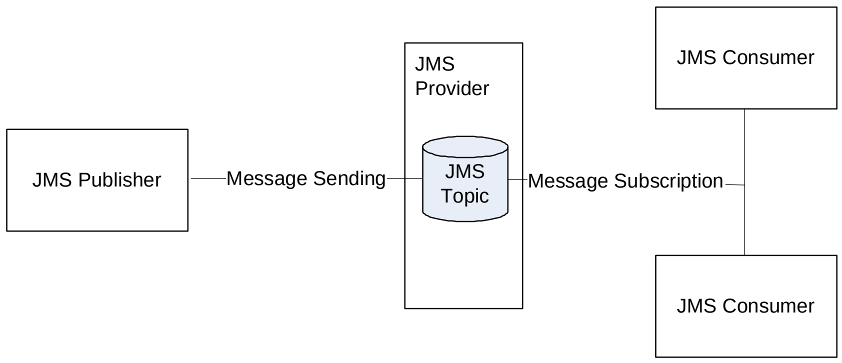


Figure 1.10: Topic (Publisher/Subscriber) model

2.4.4 Oracle Advanced Queues

The message queue systems are based on First In, First Out (FIFO) queues (with message prioritization capabilities). Oracle Advanced Queuing is implemented on databases which guarantee high availability, reliability and scalability [OSAQ].

These queues can be set as JMS queues and then benefit from the JMS interface capabilities as topic definition (multi consumer queue). JMS interface is a must have in any queue system since the interface abstracts the actual queue or topic implementation from the producing-consuming applications making possible an easier queue provider implementation change.

2.4.5 IBM WebSphere MQ Series

WebSphere MQ is the IBM implementation for a queue system. Like most of the queue systems, MQ supports JMS interface, is based on the publish-subscribe paradigm and provides extra functionalities like HyperText Transfer Protocol (HTTP) applications support with Asynchronous Javascript (AJAX) and Representational State Transfer Protocol (REST) [IBMQ].

MQ for z/OS is file based which makes any queue access more quick but it is also more difficult to have an administration access without a proprietary administration graphic user interface (GUI).

2.4.6 Oracle Application Integration Architecture

Oracle AIA was created in order to enable the development of industry specific composite business processes in a consistent manner, leveraging existing software assets that are available in IT infrastructures.

Oracle AIA includes pre built Process Integration Packs (PIPs) that use the AIA architecture to deliver composite industry processes for specific industries, using software assets from Oracle's portfolio of applications. Most of these solutions encompass orchestrated process flows, as well as pre built data integration scenarios that are meant to seamlessly connect the systems [OAIA1].

Oracle AIA addresses two types of integrations [OAIA2]:

- Functional Integration, where the several functionalities of different participating applications, exposed as services, are weaved together in the form of processes to accomplish business tasks spanning multiple applications in any enterprise;
- Data Integration, where data in one application is moved to another application.

The following concepts must be explained in order to understand what actually AIA is:

- Application Business Message (ABM) – message received or sent by a participating application;
- Enterprise Business Object (EBO) – An EBO is the canonical model for a message type to be exchange by the applications supporting the loose coupling of systems in AIA;
- Enterprise Business Message (EBM) – It’s the EBO envelope with header elements which include but is not limited to: participating applications and components, error handling backtracking and action to be performed by the EBO;
- Enterprise Business Services (EBS) - These are application and implementation independent service definitions and business level interfaces. These service definitions are implemented by all applications that participate in integration. An EBS is the foundation of an AIA PIP, facilitating distributed processing.
- Applications Business Connector Service (ABCS) – BPEL processes which are responsible to transform the ABMs into EBMs (requestors) or vice-versa (providers). The core tasks for

this type of components are: establishing connectivity with a participating application, content validation, message enrichment and transformation, population of EBM headers and invocation of the application services or EBS.

- Enterprise Business Flows (EBF) – AIA component designed to be a cross application business process responsible for content driven routing.

The following diagram shows how these AIA concepts connect or interact.

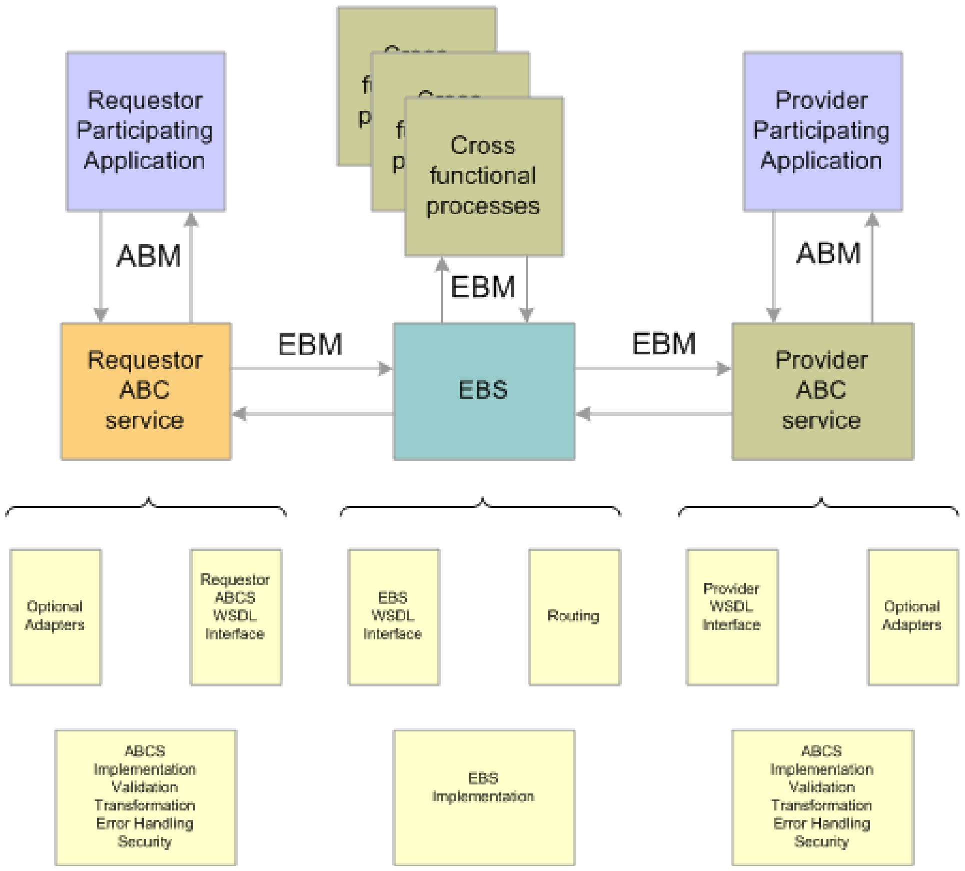

Requestor Participating Application
Cross functional processes
Provider Participating Application
ABM
EBM
ABM
Requestor ABC service
EBM
EBS
EBM
Provider ABC service
Optional Adapters
Requestor ABCS WSDL Interface
EBS WSDL Interface
Routing
Provider WSDL Interface
Optional Adapters
ABCS Implementation Validation Transformation Error Handling Security
EBS Implementation
ABCS Implementation Validation Transformation Error Handling Security

Figure 1.11:Oracle Application Integration Architecture components [OAIA1]

Oracle AIA solution foundation pack contains artifacts that are built on Oracle Fusion Middleware products, in fact, all the AIA architecture is build with the help of already existing and proven to work Oracle applications. The following diagram shows those applications.

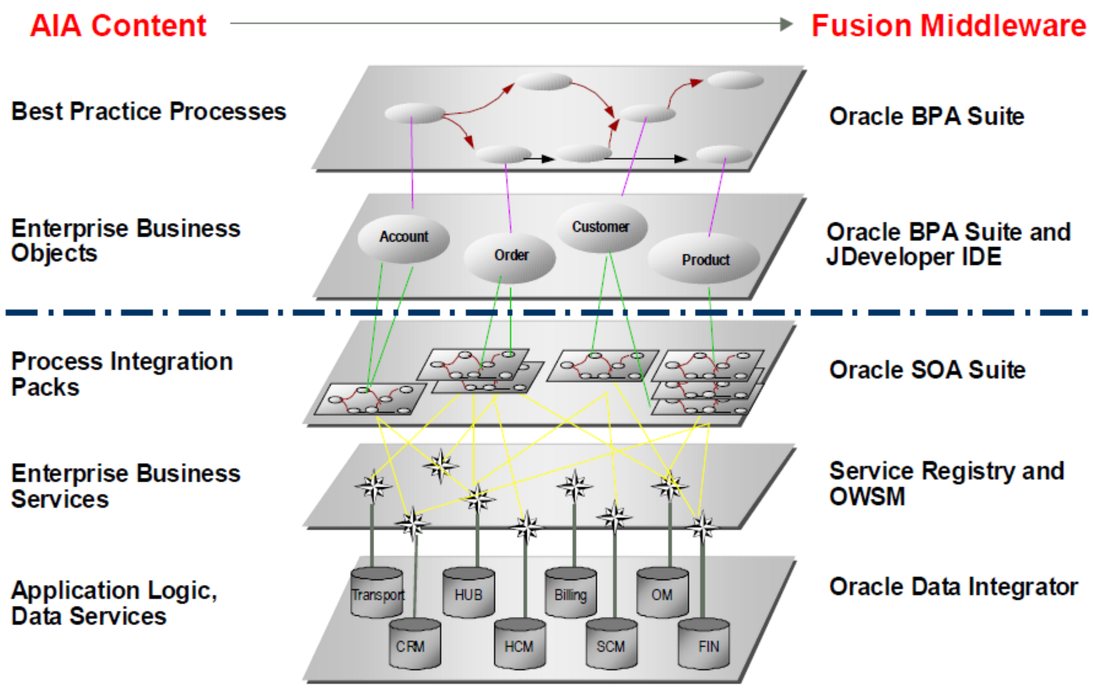

AIA Content
Fusion Middleware
Best Practice Processes
Oracle BPA Suite
Enterprise Business Objects
Account
Order
Customer
Product
Oracle BPA Suite and JDeveloper IDE
Process Integration Packs
Oracle SOA Suite
Enterprise Business Services
Service Registry and OWSM
Application Logic, Data Services
Transport
HUB
Billing
OM
CRM
HCM
SCM
FIN
Oracle Data Integrator

Figure 1.12: AIA solution artifacts

The AIA solution supports several integration styles depending on the processes needs:

Synchronous Request-Response:

This is a simple synchronized integration method that is implemented when there is a need for the master application which requests an operation on a target application (or more) to receive a response, from a simple acknowledgment message to, for example, an update data message. The only restriction on this style is that the source message application will be waiting for the reply from the target one which should be instantaneous.

Request with Delayed Response:

This style is asynchronous by nature and gives the best from the two worlds, when it’s required: gives the possibility for the target (or provider in AIA nomenclature) applications to reply to a request and the source application does not have to handle that response in the same transaction as the request but, from a process point of view, if the enterprise business flow component does not receive the response message, the user is warned as it is considered an exception.

Asynchronous Fire and Forget:

A fire and forget integration style does not give any response to the requestor application, the source does not need to know the result of the operation or it assumes that if any issue occurs, it will be dealt by the middleware or by the target application.

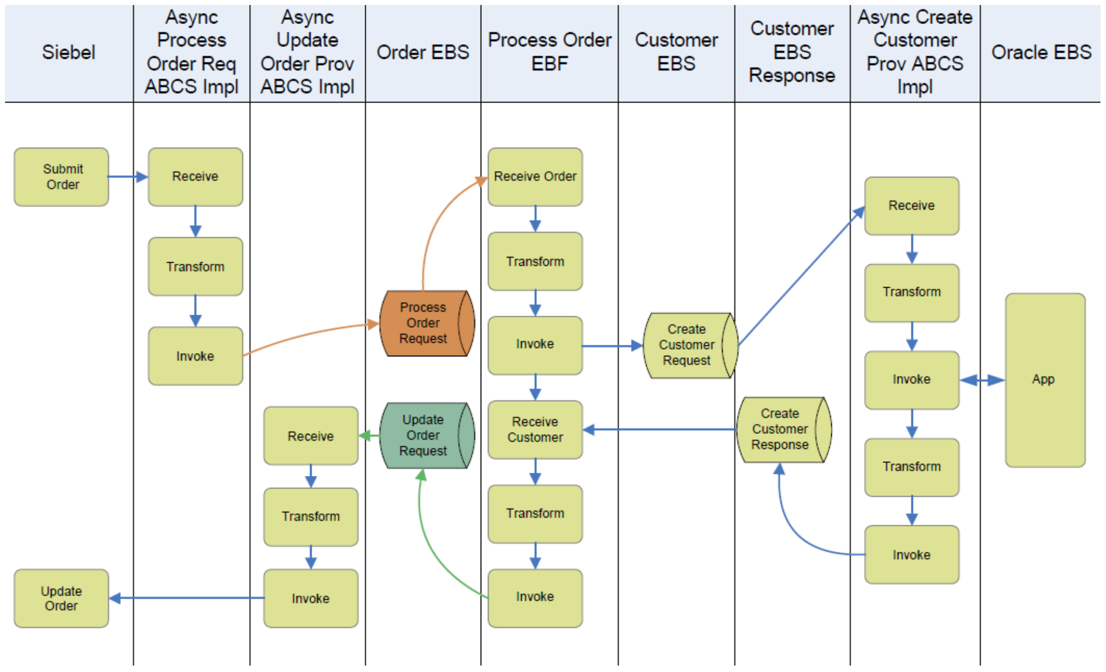


Figure 1.13:Order data interface for AIA using Fire and Forget [OAIA2]

The integration architecture for this project uses this integration style for all the data interfaces.

2.5 Tools

2.5.1 Oracle Adapters

Oracle SOA Suite includes a collection of data adapters:

- Advanced Queues (AQ) adapter;
- Database adapter;
- File adapter;
- File Transfer Protocol (FTP) adapter;
- Java Messaging System (JMS) adapter;

- Message Queue (MQ) adapter;
- Oracle Applications adapter.

All adapters support Oracle application server Java Naming and Directory Interface (JNDI) naming for connection using the application server properties.

AQ adapter, database adapter, file adapter and Oracle applications adapter also include a wizard-based resource adapter creation for the partner link (web service interface for the oracle adapters) which does not require application server properties tweaking or edition, bringing a safer testing and deployment.

Along with the connectivity properties, these adapters support Native XSD definition (nXSD) – a XML based definition for native message parsing like fixed length messages or comma separated values. Though the wizard is very limited for the normal flat file definition, it's possible to extend the parsing functionalities via manual code edition.

2.5.2 JDeveloper

Oracle has its own free Integration Development Environment (IDE) which includes full integration with every Oracle application server, database, SOA suite (BPEL, ESB, OSB, etc) and Oracle Adapters. This means that it includes UI wizards for components and connections creation, visual code edition (almost mandatory for an on-the-fly development of simple components) and version controlled deployment along with all the tools that other current IDEs as Sun's Eclipse have out-of-box.

Since it includes SOA development environment, Extensible Markup Language (XML) based development is a must have on an IDE and, from the basic XML Schema Definition (XSD) creation (Web Service Definition language – WSDL creation is also included) to the Extensible Stylesheet Language Family (XSLT) and XSLT test creations with output without requiring code deployment, JDeveloper is certainly not a limitation to SOA base development.

2.5.3 Oracle iProjects Files

iProjects Files is part of the Oracle Collaboration Suite for projects and it has been used on this programme as a documentation archive.

This Oracle extranet is web based, includes file system integration with client's operating system and version control. Every user can have its own workspace and external parties, such as a company, can own a public or private workspace to be used by the assigned users. The best functionality of this application is the direct URL link to an uploaded file and the biggest issue found was the timeouts and server availability which sometimes can be critical.

2.6 Conclusions

All the technologies described on this chapter were used on this project. For example, the components for this data integration are BPEL processes (using Oracle SOA suite BPEL process manager) that were developed using JDeveloper and connect to the applications using Oracle adapters.

The following diagram describes how these applications are present on the project.

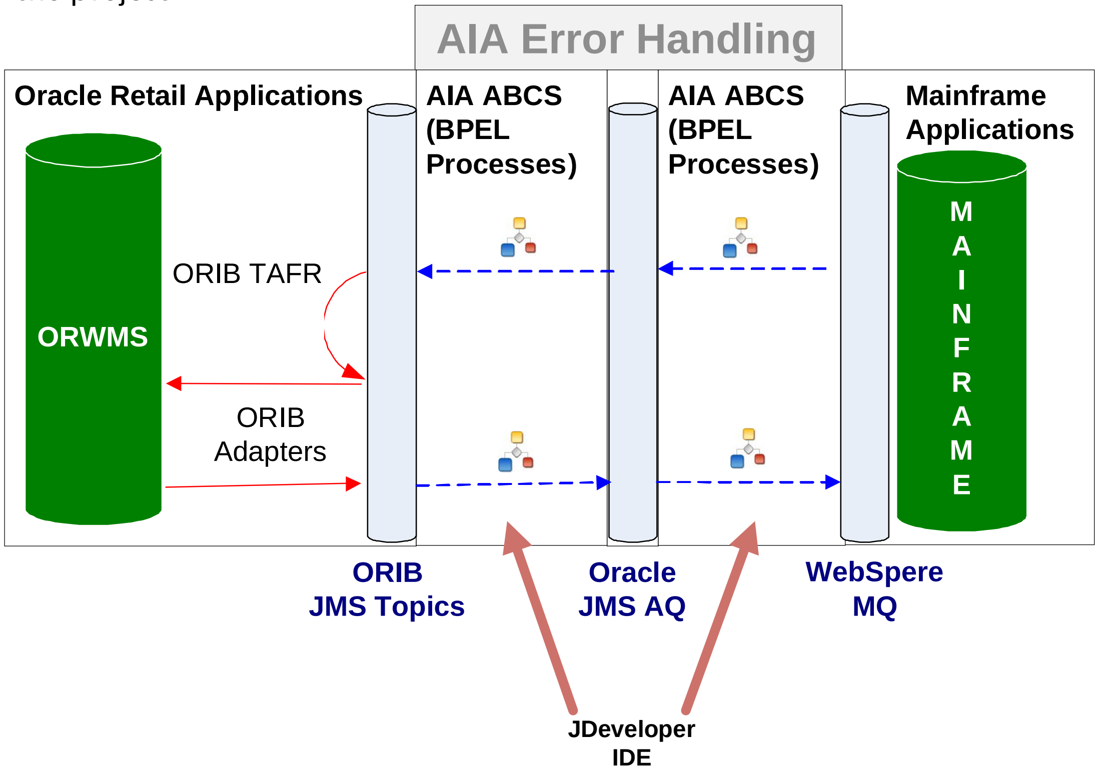


Figure 1.14:Project technologies overview

The way that these technologies interact is described on the next chapter.

Chapter 3

Systems Integration using Oracle SOA Suite

This chapter describes the project at a functional level starting with an introduction to the type of client that requested this service, the as-is and to-be architecture. It also describes the project's plan on a macro and detailed levels, followed by the functional requirements, integration scenarios and constraints ending with a complete description of the solution at the functional level including non functional solutions as messaging order and error handling.

3.1 Introduction

3.1.1 Client

This project's client is among the top 3 retailers of the United Kingdom, and among the top 50 of the world. The average sales income for the top 3 retailers in the United Kingdom is 35,66M€ in 2008 which is an indicator for the size of this business area or industry.

With these kind of numbers and knowing that this retailer wants to renew almost completely its business supporting applications (from financials, warehouse and trading to the end point of sale) with Oracle products if possible, it was expected to have a system implementation with high demand, a future system with high availability and transactional effort required and a long road ahead, since this programme is expected to last at least 5 years.

3.1.2 As Is High Level Architecture

The client initial solution for logistics processes is based on two systems: Mainframe and Swisslog’s Warehouse Manager. These two systems are integrated via basic file transfers and integration functionalities such as message ordering or error handling are solved using a batch scheduler for the file transfer for each existing interface.

3.1.3 To Be High Level Architecture

Though the initial (decided during the proof of concept analysis) methodology and design approach was based on simplicity and a quick implementation was required, the final proposed architecture and components involved on this integration wasn’t, due to the decision of Oracle AIA as the framework for the all the integration components in Oracle applications and middleware implementation on this client. The following diagram represents the architecture of the integration system for this project, with applications and touch point involved, for a complete overview.

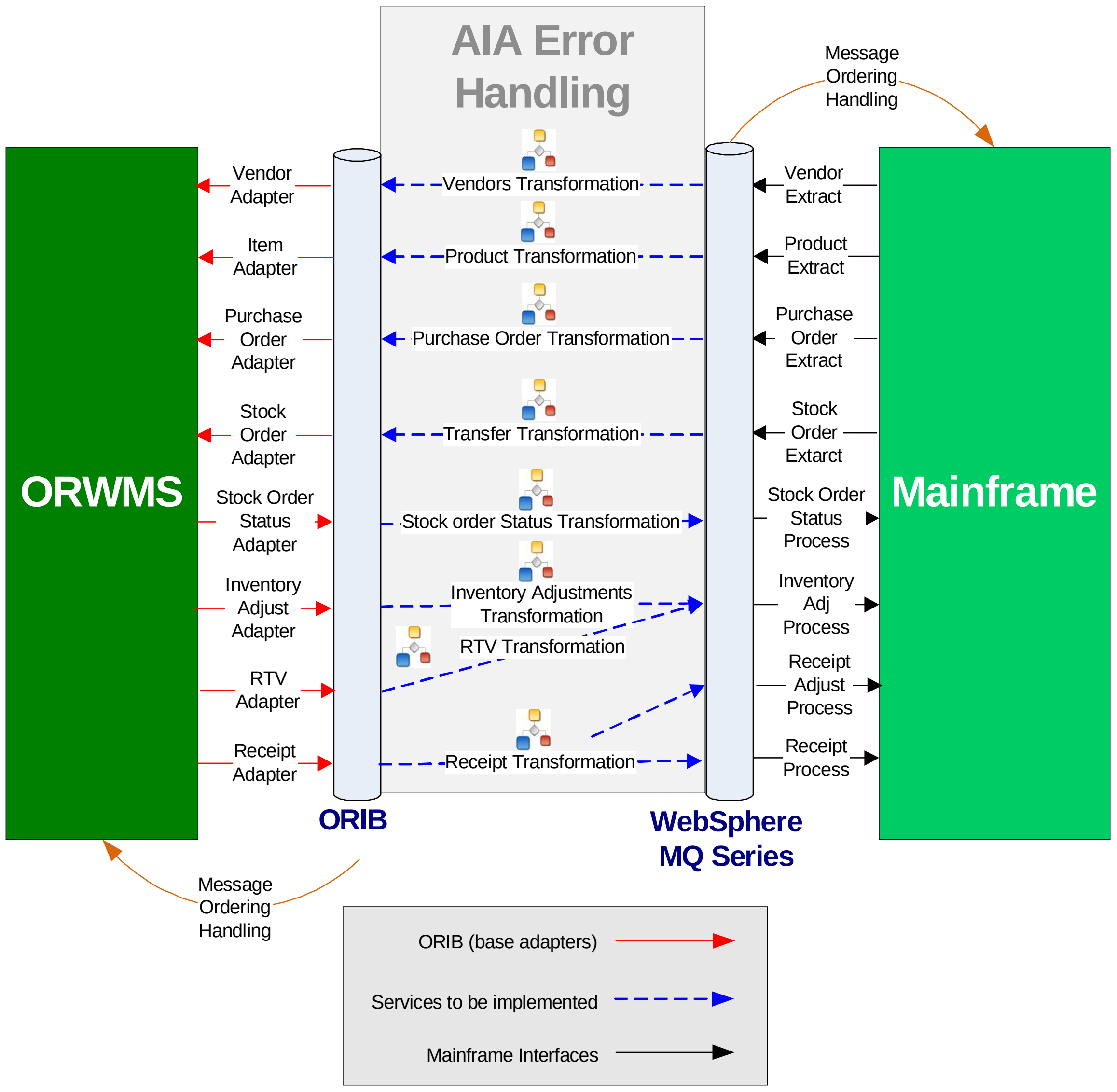


Figure 1.15:To-Be high level architecture

This diagram will be explained and detailed in the following sections.

3.2 Planning

3.2.1 Transition Plan

Since this project is part of a programme with at least nineteen projects for the same client, with different timetables and some depend on others like this one, it’s necessary to give an overview of the programme to have a time reference and scope.

The project starts in the middle of 2008 and have the rollout in 2009's Q4, for a complete ORWMS instance running, integrated with Mainframe for frozen warehouses (only one planned as a pilot). The next steps will be to gradually replace Mainframe with ORMS when the trading projects succeed and are capable of full and standalone Oracle Retail integration.

The following diagram completes the information given.

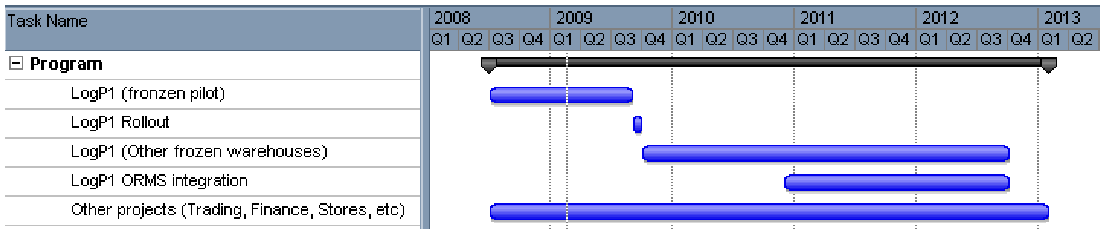
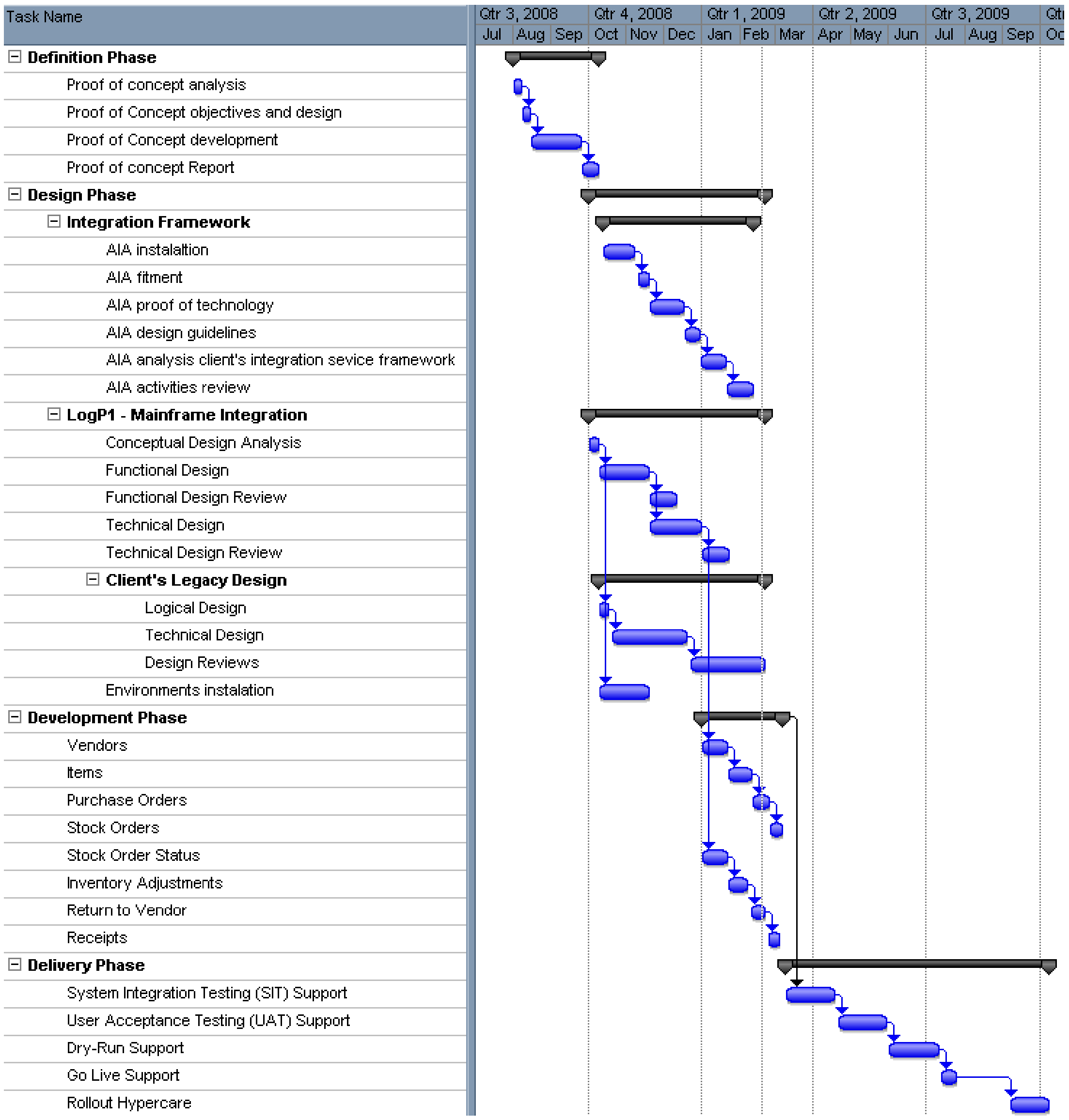


Figure 1.16:Gant chart for programme project overview

3.2.2 Project Planning

The project from the proof of concept until the go-live will take about one year but since this report is based on the author's work, it is relevant to present the full Logistics Project 1 (LogP1) Mainframe Integration process and deliverables timeline which is shown on the following Gantt chart.



Task Name
⊟ **Definition Phase**
Proof of concept analysis
Proof of Concept objectives and design
Proof of Concept development
Proof of concept Report
⊟ **Design Phase**
⊟ **Integration Framework**
AIA instalaltion
AIA fitment
AIA proof of technology
AIA design guidelines
AIA analysis client's integration sevice framework
AIA activities review
⊟ **LogP1 - Mainframe Integration**
Conceptual Design Analysis
Functional Design
Functional Design Review
Technical Design
Technical Design Review
⊟ **Client's Legacy Design**
Logical Design
Technical Design
Design Reviews
Environments instalation
⊟ **Development Phase**
Vendors
Items
Purchase Orders
Stock Orders
Stock Order Status
Inventory Adjustments
Return to Vendor
Receipts
⊟ **Delivery Phase**
System Integration Testing (SIT) Support
User Acceptance Testing (UAT) Support
Dry-Run Support
Go Live Support
Rollout Hypercare

Figure 1.17:LogP1 – Mainframe integration Gantt chart

The following table presents the resources allocated for the tasks previously show.

Table 1.2: Project tasks and resources

Task Name	Resources
Definition (PoC)	Author
Integration Framework	Wipro Integration Offshore Team
Conceptual Design Analysis	Author
Functional Design	Author
Technical Design	Author
Client’s Legacy Design	Client’s Heritage Integration Team
Design Review	Client’s Business Application Group; Design Authority Group; Architecture Group

Development Phase	Wipro Integration Offshore Team; Author
SIT and UAT Support	Wipro Testing Team; Author
Dry-run Support	Wipro Testing Team; Author; Client's Business Application Management
Go Live Suport	Wipro Testing Team; Author; Client's Business Application Management
Rollout Hypercare	Wipro Testing Team; Author; Client's Business Application Management; Client's Support and Helpdesk

Each interface developments include:

- Component developments done by the offshore team and supported by the author;
- Two internal unit testing cycles at offshore;
- Component integration testing with Mainframe processes and ORIB interfaces done by the author.

Since the testing, delivering and support of the developments will happen in the future, the timeline shown is a reference and can be changed.

3.2.3 Methodology

The deliverable methodology for this programme which has been defined between Wipro Technologies and the client, has the following structure for each project:

1. Conference room pilot (CRP) workshops (business streams only);
2. CRP report (business streams only);
3. Requirements Definition document (RDD);
4. Conceptual Design definition and Show & Tell (presentation);
5. Functional Design document (FDD) and Show & Tell;
6. Technical Design document (TDD) and Show & Tell;
7. Development phase;
8. Testing phase;
9. Pre-production phase;
10. Go live phase;
11. Rollout Phase;

Each document is signed off by the client's business applications managers which delegate reviewers for each one between design authority groups, business experts and Oracle consultants. The sign off process which includes reviews and changes tracking has very strict guidelines that can guarantee process conclusion since the high number of stakeholders involved is a risk to project continuity.

3.3 Requirement and Constraints

3.3.1 Functional Requirements

As defined in the methodology section of this chapter, any integration project or data interface is requested, firstly, by the business, during CRPs which will be translated, after the CRP report, into a requirements definition document for integration, which, as the name explains, defines which are the requirements for each interface and which should be set as constraint for the further requirements depuration. The following table gives an overview for the data interface requirements for this project.

Table 1.3: Summary of Mainframe Integration data flows

Data Flow ID	Entity	Source	Target	Frequency	Type
DF002	Vendor	Legacy System	ORWMS	Run Times: Mon – Wed: 07.44, 09.30, 13.00, 16.00, 20.02 Thu – Fri: 07.44, 09.30, 20.02 Sat: 07.44, 11.45 Sun: 07.44	Batch
DF003	Items	Legacy System	ORWMS	Every 15 mins	Batch
DF004	Purchase Order	Legacy System	ORWMS	Every 15 mins	Batch
DF005	Stock Order	Legacy System	ORWMS	Every 15 mins	Batch
DF006	Stock Order Status	ORWMS	Legacy System	Near Real Time	Near Real Time
DF007	Inventory Adjustments and RTV	ORWMS	Legacy System	Near Real Time	Near Real Time
DF008	Receipt	ORWMS	Legacy System	Near Real Time	Near Real Time

In data interfaces specification, only one participating application can be the master or owner of the data, since the raw data must be created on an application via a GUI or via application processes. The following table presents the ownership requirements for the data interfaces previously specified.

Table 1.4: Data ownership

Data Flow ID	Master	Slave
DF002 - Vendor	Mainframe	ORWMS

DF003 - Items	Mainframe	ORWMS
DF004 – Purchase Order	Mainframe	ORWMS
DF005 – Stock Order	Mainframe	ORWMS
DF006 – Stock Order Status	ORWMS	Mainframe
DF007 – Inv Adjustments	ORWMS	Mainframe
DF007 – RTV	ORWMS	Mainframe
DF008 – Receipt	ORWMS	Mainframe

The data interfaces purpose it to exchange messages and each message corresponds to an action on the receiving application. The following table presents the message actions requirements for each interface according to the application needs and identified during the CRP sessions.

Table 1.5: Data interface's message types

Data Interface	Message Types
Vendors	Creation Address modification
Items	Creation Header modification Supplier creation Supplier country modification Supplier country dimensions modification UPC creation UPC modification UPC deletion
Purchase Orders	Creation Header modification
Stock Orders	Creation Header modification Detail deletion
Stock Order Status	Creation
Inventory Adjustments	Creation
RTV	Creation
Receipts	Creation Modification

3.3.2 Functional Scenarios

Each data interface and corresponding message types or actions were identified according to business action scenarios that are presented on the following tables.

Table 1.6: Integration scenarios for vendors

Scenario	Message sent to ORWMS	Update on ORWMS
New supplier created on Mainframe	Supplier creation	Vendor info Vendor address info
Supplier address amended on Mainframe (ammendment to any of the 3 address fields)	Supplier address modification	Vendor address info

Table 1.7: Integration scenarios for items

Scenario	Message sent to ORWMS	Update on ORWMS
New product created on Mainframe	Item creation (including master, item supplier, item supplier country, item dimensions and UPC – with EAN and 3 OCCs)	Item info according to the message sent
EAN or OCC added to produ ct on Mainframe	Header modification and item UPC creation	Item master and item UPC info
Change on the primary to an EAN or a OCC	Header and Item UPC modification	Item master and item UPC info
EAN or OCC deleted from product on Mainframe	Header modificati on and UPC deletion	Item master and item UPC info
Product description change on Mainframe	Header modification	Item master info
Product marked as price marked	Header modification	Item master info
Product unmarked as price marked	Header modification	Item master info
Product case pack change on Mainframe	Item supplier country modification	Item supplier country info
Product ti-hi change	Item supplier country modification	Item supplier country info
Case dimension data updated on Mainframe	Item supplier country dimentions (ISCDim) modification	ISCDim info
Supplier for the product updated	Item supplier creation	Item supplier, supplier country and ISC dimentions info
Item discontinued or deleted	-	Data not communicated to ORWMS

Table 1.8: Integration scenarios for purchase orders

Scenario	Message sent to ORWMS	Update on ORWMS
New purchase order from supplier created on Mainframe	Purchase Order header and detail creation	PO and PO details info
Added detail for a Purchase Order	-	Not valid
Deleted detail	-	Not valid
Failed delivery and PO rescheduled on Mainframe	New PO and PO detail creation	PO and PO details info
Oder received and PO close	PO header modification	PO info with cancelled status

Table 1.9: Integration scenarios for stock orders

Scenario	Message sent to ORWMS	Update on ORWMS
New stock order created via allocations order program on Mainframe	Stock Order header and detail creation	Stock order and stock allocation info
New stock order created via canteen order program on Mainframe	Stock Order header and detail creation	Stock order and stock allocation info
New stock order created via disk order program on Mainframe	Stock Order header and detail creation	Stock order and stock allocation info
New stock order created via allocation SMS order program on Mainframe	Stock Order header and detail creation	Stock order and stock allocation info

Note: Order type is differentiated via the first letter of the order code.

Table 1.10: Integration scenarios for stock order status

Scenario	Message sent to Mainframe	Update on Mainframe
Stock Order from allocations program expires on ORWMS	Stock Order status with expired status	Stock on Hand (SoH) info updated and store credited
Stock Order from canteen order program expires on ORWMS	Stock Order status with expired status	SoH info updated and store credited
Stock Order from disk order program expires on ORWMS	Stock Order status with expired status	SoH info updated and store credited
Stock Order from allocations SMS program expires on ORWMS	Stock Order status with expired status	SoH info updated and store credited

Table 1.11: Integration scenarios for inventory adjustments and RTV

Scenario	Message sent to Mainframe	Update on Mainframe
Physical positive inventory adjustment	Inventory adjustment creation	SoH figure is increased with the stock adjustment quantity
Physical negative inventory adjustment	Inventory adjustment creation	SoH figure is decreased with the stock adjustment quantity
Non-physical positive adjustment (i.e. taking stock out of unavailable state)	Inventory adjustment creation	SoH figure is increased with the stock adjustment quantity
Non-physical negative adjustment (i.e. putting stock out of unavailable state)	Inventory adjustment creation	SoH figure is decreased with the stock adjustment quantity
Non-physical positive adjustment and physical negative adjustment	Inventory adjustment creation	SoH figure is decreased with the stock adjustment quantity
Stock to return to supplier	RTV creation	SoH figure is decreased with the stock adjustment quantity
Troubled stock to return to supplier	RTV creation	SoH figure is decreased with the stock adjustment quantity

Table 1.12: Integration scenarios for Receipts

Scenario	Message sent to Mainframe	Update on Mainframe
Stock received on an Appointment in ORWMS and appointment is closed	Receipt and receipt detail creation	Item SoH figure changed to include receipted quantity
Stock received on an appointment and troubled at the pallet level in ORWMS and the appointment is closed	Receipt and receipt detail creation	Item SoH figure changed to include receipted quantity
Stock subsequently adjusted post receipt	Receipt (adjustment type) and receipt detail creation	Item SoH figure changed to include receipted quantity
Stock subsequently adjusted post receipt and is troubled	Receipt (adjustment type)and receipt detail creation	Item SoH figure changed to include receipted quantity

3.3.3 Functional Decisions

The following decisions were taken by the author, the integration team and the client's architecture group on the functional design phase:

- All data flows will be running in a near-real-time frequency, from ORIB to WebSphere MQ and vice-versa;
- Oracle SOA Suite BPEL process manager and Oracle Advance Queues are used to intermediate ORIB and Mainframe;
- The integration components required between WebSphere MQ and Mainframe are out of the author’s functional documentation scope since it’s part of the client’s heritage integration team (HIT) responsibilities;
- ABCS components will post and receive fixed length messages from and to the Mainframe WebSphere MQ queues in the agreed format between the author and the client’s HIT;
- The interfaces described should only include data for frozen warehouses. Due to ORWMS data constraints, this integration does not support data for ambient and fresh warehouses, except for suppliers which will all be sent to ORWMS;
- This integration architecture supports more than one ORWMS instance (for frozen warehouses) using ORIB routing functionality;
- ORIB version is 13.0.2.1 and Oracle AIA version is 2.2.1 Foundation Pack due to ORIB message schemas definition and AIA EBM header and fault schemas definition.

3.3.4 Responsibilities and Work Boundaries

The following diagram represents the project teams’ responsibilities for development, support and maintenance for each participating layer for this integration.

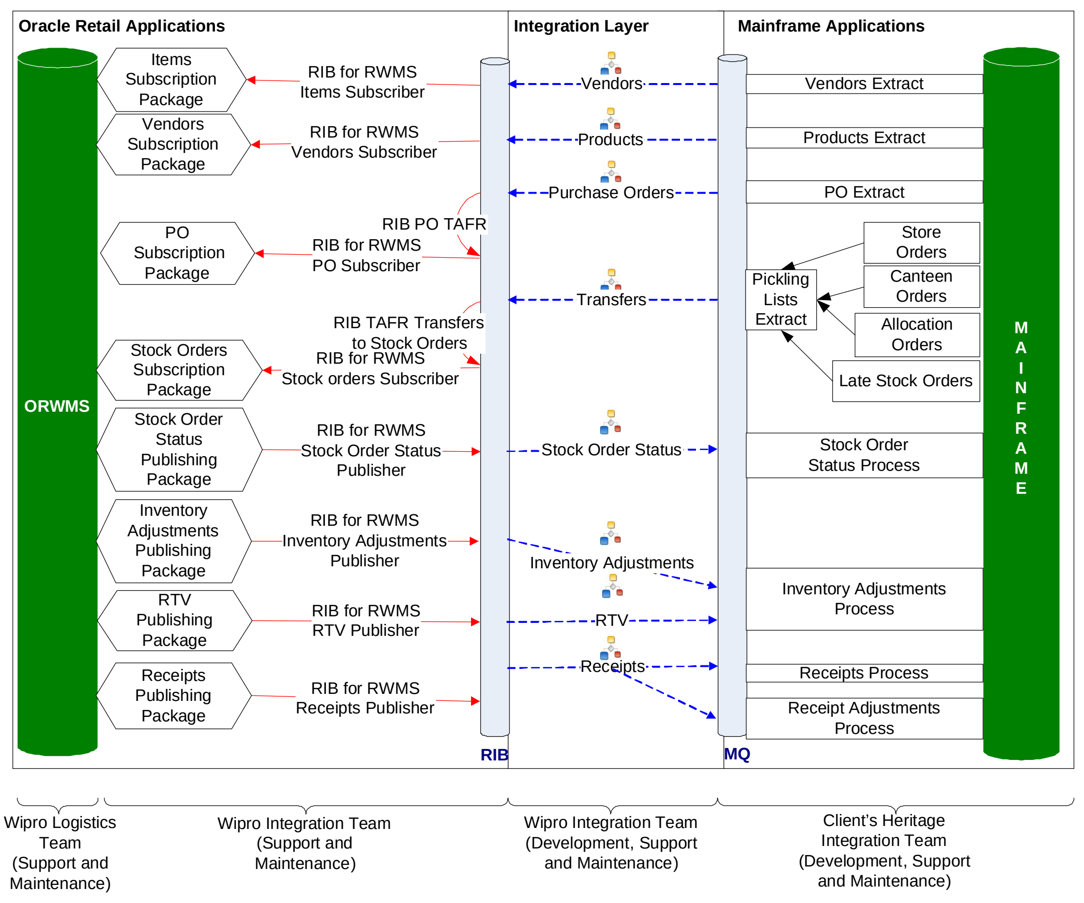

Oracle Retail Applications
Integration Layer
Mainframe Applications
ORWMS
Items Subscription Package
Vendors Subscription Package
PO Subscription Package
Stock Orders Subscription Package
Stock Order Status Publishing Package
Inventory Adjustments Publishing Package
RTV Publishing Package
Receipts Publishing Package
RIB for RWMS Items Subscriber
RIB for RWMS Vendors Subscriber
RIB PO TAFR
RIB for RWMS PO Subscriber
RIB TAFR Transfers to Stock Orders
RIB for RWMS Stock orders Subscriber
RIB for RWMS Stock Order Status Publisher
RIB for RWMS Inventory Adjustments Publisher
RIB for RWMS RTV Publisher
RIB for RWMS Receipts Publisher
Vendors
Products
Purchase Orders
Transfers
Stock Order Status
Inventory Adjustments
RTV
Receipts
RIB
MQ
Vendors Extract
Products Extract
PO Extract
Store Orders
Canteen Orders
Allocation Orders
Late Stock Orders
Pickling Lists Extract
Stock Order Status Process
Inventory Adjustments Process
Receipts Process
Receipt Adjustments Process
MAINFRAME
Wipro Logistics Team (Support and Maintenance)
Wipro Integration Team (Support and Maintenance)
Wipro Integration Team (Development, Support and Maintenance)
Client's Heritage Integration Team (Development, Support and Maintenance)

Figure 1.18:Responsibilities and work boundaries

3.4 System Integration between ORWMS and Mainframe

3.4.1 Application Touch Points

Touch point, from an integration component development point of view, is the source or target from which the developed components get or put the messages that are translated into database data. Since this integration includes two applications, there are two touch points defined.

Mainframe

The first approach to connect the required components to be developed with Mainframe was to use Oracle FTP adapter, since Mainframe is file based and it does not require third party components installation on Mainframe servers but, since Oracle FTP adapter is not certified to work with a z/OS because it lacks FTP functionalities, it has been decided to install WebSphere MQ Series for Mainframe, providing a message queue system as touch point for Mainframe, which is better from a message-oriented middleware integration point of view.

A quick proof of concept connecting an Oracle ESB component and then BPEL component using an Oracle MQ adapter and an Oracle JMS adapter with Mainframe's WebSphere MQ Series queue was done in order to prove connectivity. Unfortunately, the way WebSphere MQ Series was installed on Mainframe did not permit a JMS interface, therefore Oracle MQ adapter was chosen as touch point publisher and consumer for Mainframe.

ORWMS

The touch point for ORWMS is not part of this system's architecture. ORIB topics, proven to be the best touch points for ORWMS near real time integration, was chosen since it guarantees all the non functional requirements for a database touch point: message order, error logging, handling and recovering. This decision became final when all the data interfaces required for this project were identified in ORIB base package. ORIB is based on JMS topics which can be accessed by external applications but does not guarantee application level clustering.

3.4.2 Message Types and Definitions

This integration is based on a messageoriented middleware integration which means that the integration components will exchange messages between them.

There are three different message types to be exchanged:

- ORIB messages, ABMs;
- Retail Business Messages (RBM) between requestors and providers ABCS and;
- Fixed Length Messages (FLM) from or to Mainframe WebSphere MQ.

ORIB Messages

ORIB Messages, or RIB Messages for short, are XML messages which contain the ORIB payloads (business messages) escaped on the message data field of an envelope. Each RIBMessages XML message envelope can contain more that one RIBMesssage message and each RIBMessage can contain only one ORIB payload.

The envelope for each RIBMessage contains several information for the ORIB adapters such as message family, message action, message identification, routing details, message recovering details to ORIB hospital, etc.

The next figure presents the RIBMessages schema definition.

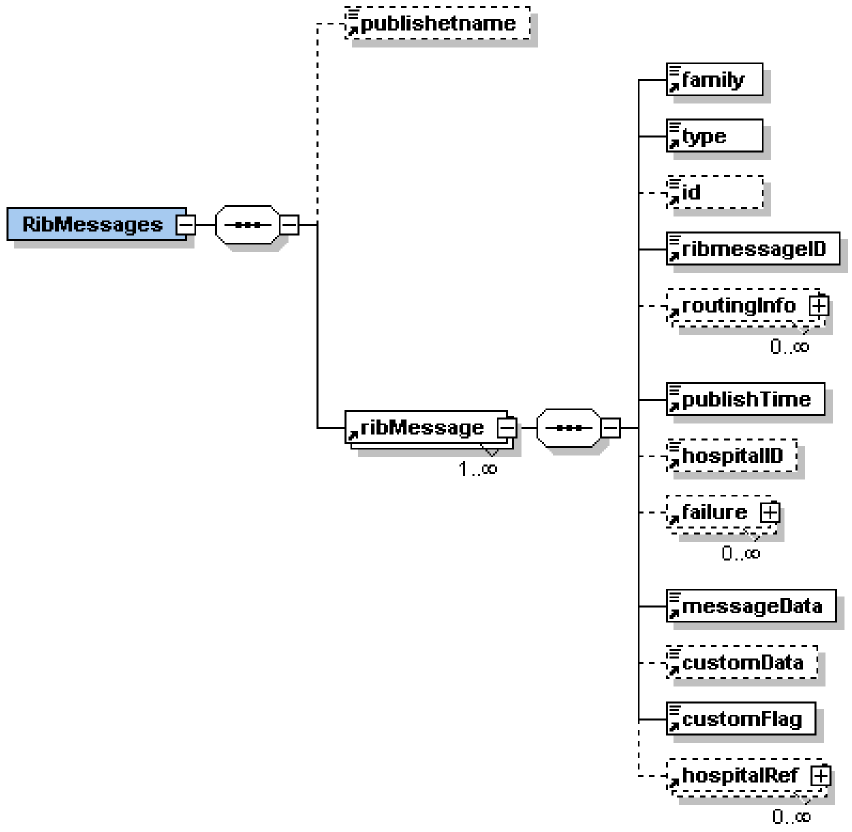


Figure 1.19:RIB messages schema definition

Each RIB payload contains header and details about each data interface, description for each creation, modification and reference data for each deletion.

Retail Business Messages

In order to have interface level scalability, it has been decided that the messages should be transformed in to a type of canonical message before they are transformed into the target ABMs. The canonical message is defined on Oracle AIA as Enterprise Business Message (EBM) but, since this project will not use all the functionalities of the Oracle AIA framework, such as message type registration, it has been decided that the model can be simplified into a EBM header and envelope and that the data area should be switched from the canonical EBOs into the RIB payloads.

The following diagram represents an example of a schema for a Retail Business Message (simplified EBMs) for the product (or, in Oracle's terminology, item) data type:

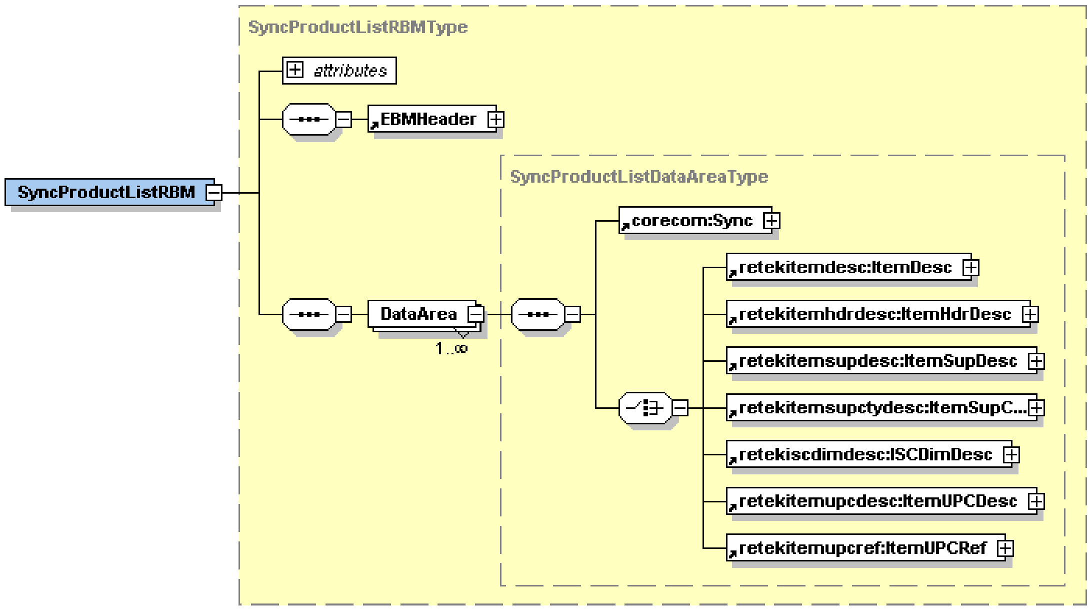

SyncProductListRBMType
attributes
EBMHeader
SyncProductListRBM
SyncProductListDataAreaType
corecom:Sync
retekitemdesc:ItemDesc
retekitemhdrdesc:ItemHdrDesc
retekitemsupdesc:ItemSupDesc
retekitemsupctydesc:ItemSupC...
retekiscdimdesc:ISCDimDesc
retekitemupcdesc:ItemUPCDesc
retekitemupcref:ItemUPCRef
DataArea
1..∞

Figure 1.20:RBM schema example

The EBM header will be used to perform message logging, error handling, logging and notification using Oracle's AIA error handling framework.

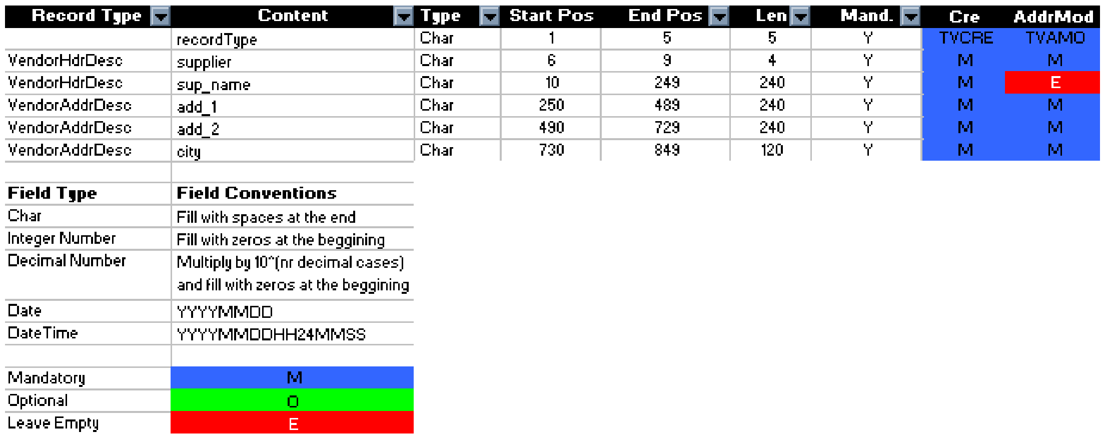
Oracle AIA supports different verbs (or action codes) like create, update, delete, query, sync, etc, and it has been decided that all the RBM verbs should be sync for this project, since the messages' purpose isn't actually to give an order to the slave application to create, modify or delete an object but instead is to inform the slave application that the source data has changed and to synchronize it.

Mainframe Fixed Length Messages

The Mainframe file system and data physical model is based on files with fixed length fields. Since Oracle adapters support the parsing of this type of files into XML messages (which was part of the connectivity proof of concept for this project), Mainframe will receive and publish those type of messages from or to the touch point queues.

As a side note, Oracle SOA Suite (and BPEL process manager) is defaulted to exchange UTF-8 encoding messages and, since mainframe is based on the EBCDIC/cp500 encoding type, the nXSD should be set to that encoding, or else, if there is any special characters constraint, a converter utility must be used on the Mainframe side as a post or pre process for the queues publishing or subscribing.

The format for each data interface has been decided between the client and the author, being maintained by the author. The following table presents a representation of this format definition for vendors.



Record Type	Content	Type	Start Pos	End Pos	Len	Mand.	Cre	AddrMod
	recordType	Char	1	5	5	Y	TVCRE	TVAMO
VendorHdrDesc	supplier	Char	6	9	4	Y	M	M
VendorHdrDesc	sup_name	Char	10	249	240	Y	M	E
VendorAddrDesc	add_1	Char	250	489	240	Y	M	M
VendorAddrDesc	add_2	Char	490	729	240	Y	M	M
VendorAddrDesc	city	Char	730	849	120	Y	M	M

Field Type	Field Conventions
Char	Fill with spaces at the end
Integer Number	Fill with zeros at the beggining
Decimal Number	Multiply by 10^(nr decimal cases) and fill with zeros at the beggining
Date	YYYYMMDD
DateTime	YYYYMMDDHH24MMSS

Mandatory	M
Optional	O
Leave Empty	E

Figure 1.21:Fixed length message definition example for vendors

3.4.3 Functional Architecture

All the data interfaces will follow the presented functional and general architecture, in order to cope with AIA design conventions.

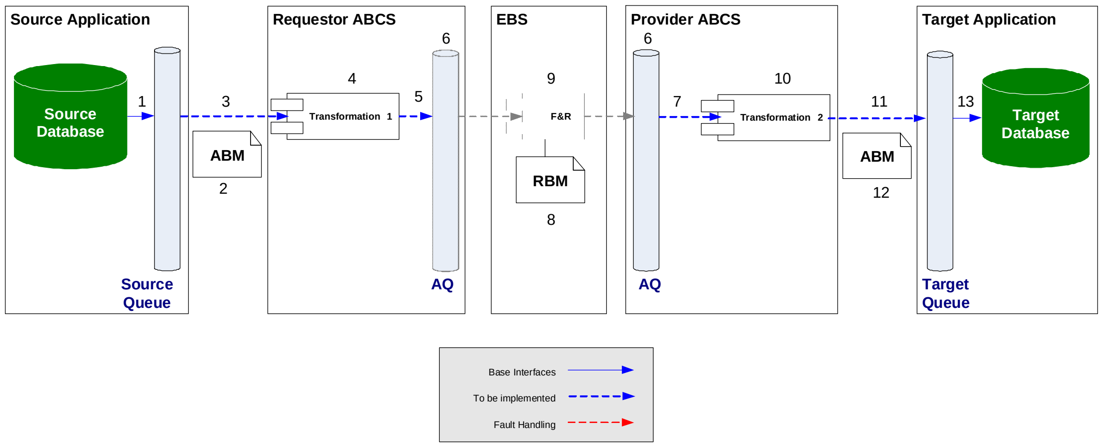


Figure 1.22:Functional architecture

Source Application

1. Integration between the source databases and source queues or topics is considered to be part of the source application.

Requestor ABM

2. The message types between the source application and Requestor ABCS (which are transformation and forwarding components) are referred as requestor ABMs;
3. Oracle Adapters (JMS or MQ) will be used to consume messages from the source queues or topics;

Requestor ABCS

4. Requestor ABCS should transform the ABMs into RBMs and should be developed using Oracle BPEL;

Middle AQ

5. Oracle JMS Adapters will be used to publish messages from the requestor queues or topics into the AQ;
6. Source ABCS should publish RBMs into a middle queue system and target ABCS should consume from that system in order to decouple this architecture and to permit future implementations of routing components;
7. Oracle JMS Adapters will be used to consume messages from the AQ;

RBM

8. These are RIB payloads with EBM header

EBS

9. EBS module will not be required but the design and implementation should be extensible to support this architecture, in order to support future requirements on context and/or content based routing;

Provider ABCS

10. Provider ABCS should transform the RBMs into target ABMs and should be developed using BPEL;

Provider ABM

11. Oracle Adapters (JMS or MQ) will be used to publish messages into the target queues or topics;
12. The message types between the Provider ABCS and the target application are referred as provider ABMs;

Target Application

13. Integration between the target databases and target queues is considered to be part of the source application.

The following diagram describes an end to end functional overview of the integration architecture between ORWMS and Mainframe.

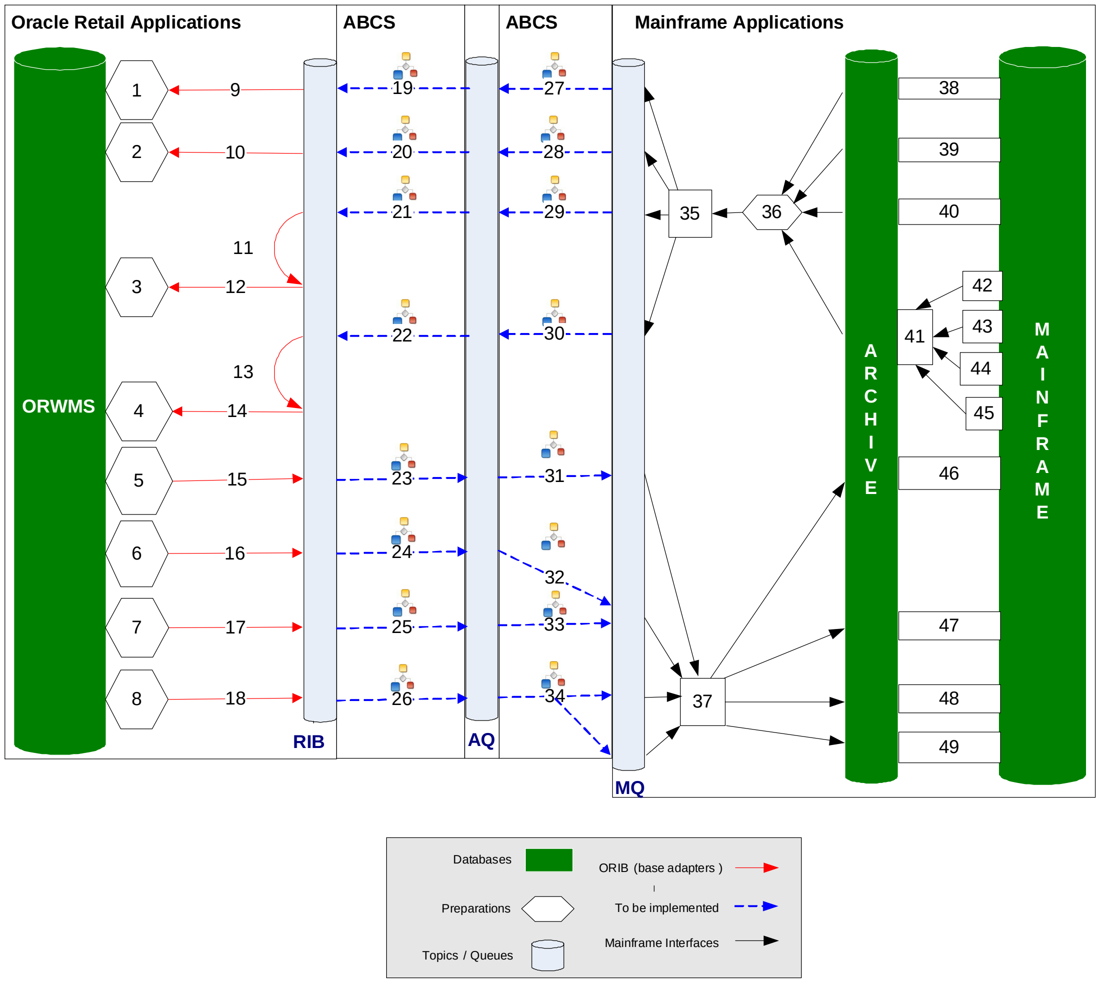


Figure 1.23: Component functional architecture

Subscription packages: 1. Items

2. Vendors
3. Purchase Orders
4. Stock Orders

Publishing packages:

5. Stock Order Status
6. Inventory Adjustments
7. Return to Vendor
8. Receipts

ORIB adapters:

9. Items subscriber
10. Vendors subscriber
11. Purchase Order TAFR
12. Purchase Order subscriber
13. Transfers to Stock Orders TAFR
14. Stock Orders subscriber
15. Stock Order Status publisher
16. Inventory Adjustments publisher
17. Return to Vendor publisher
18. Receipts publisher

Application Business Connector Services Mainframe providers:

19. Products
20. Vendors
21. Transfers
22. Stock Orders

Application Business Connector Services ORIB requestors:

23. Stock Order Status
24. Inventory Adjustments
25. Return to Vendor
26. Receipts

Application Business Connector Services Mainframe requestors:

27. Products
28. Vendors
29. Transfers
30. Stock Orders

Application Business Connector Services ORIB providers

31. Stock order Status
32. Inventory Adjustments
33. Return to Vendor
34. Receipts

WebSphere MQ Series Mainframe utilities:

35. Put Message from File
36. File Pre-process
37. Get Message to File

Mainframe's extract programs:

38. Vendors
39. Products
40. Picking Lists
41. Store Orders
42. Canteen Orders
43. Allocation Orders
44. Late Stock Orders

Mainframe's process programs

45. Stock Order Status
46. Inventory Adjustments
47. Receipts
48. Receipt Adjustments

3.4.4 Component Descriptions

Subscription Packages

These are PL/SQL packages standing on ORWMS database though they are part of the ORIB architecture. These packages receive Oracle objects and perform the table's insertion, update or delete according to them. This integration implementation does not include modifications to these packages.

Publishing Packages

These are PL/SQL packages standing on ORWMS database though they are part of the ORIB architecture. These packages consume data from staging tables and transform it into Oracle objects. This integration implementation does not include modifications to these packages.

ORIB adapters

ORIB PL/SQL adapters are EJBs or MDBs that listen to the staging tables or to the ORIB topics and perform the messaging forwarding, transformation or routing. This integration implementation does not include modifications to these adapters.

Mainframe requestors

Mainframe requestors consume fixed length messages from the Mainframe queues and publish them in the middle queues, transforming them into RBMs.

Mainframe providers

A mainframe provider transform RBMs consumed from the middle queues into RIBMessages and publishes the transformed messages on the ORIB JMS topics.

ORIB Requestors

These ABCS get the ORIB ABMs (RIBMessages) from the ORIB JMS topics and transform them into RBMs to be published on the middle queues.

ORIB providers

These ABCS get the RBMs from the middle queues and transform them into fixed length messages to be published on the Mainframe queues.

3.4.5 Error Handling, Notification and Logging

Error logging will be done using the Logging functionality provided as part of AIA error handling framework. The logging component writes the error logs to a log file which can be viewed using the Enterprise Manager console of Oracle SOA suite.

Error notification will be done using the notification functionality provided as part of AIA error handling framework. The notification is send to the recipient list defined at service level through the AIA console. If recipients are not defined at the service level, notification will be sent to the default mailing list.

All BPEL processes with faulted instances can be viewed through the BPEL console. Depending on the type of fault, the instance could either be ‘waiting for input’ or appear as ‘faulted’. For instances waiting for input, the concerned user can perform one of the following:

- Modify or correct the payload and resume or restart the faulted BPEL service instance;
- Resume or restart the faulted BPEL process instance without modifying the payload;
- Terminate the faulted BPEL service instance

For instances that appear faulted, the above user action can not be performed. However the BPEL console can be used to view the error details.

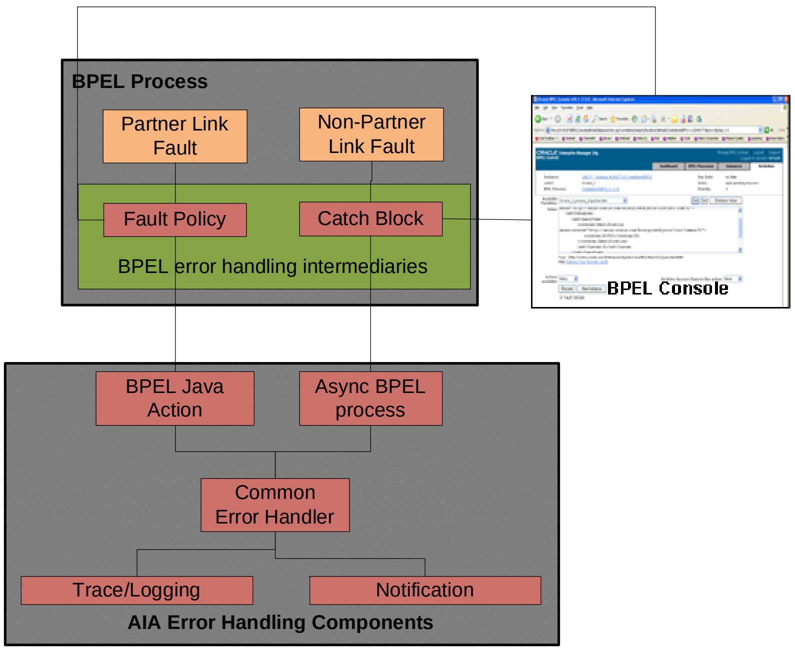


Figure 1.24:Oracle BPEL and AIA error handling integration

On occurrence of a fault, the BPEL Fault Management framework decides if it is a Partner Link or non-Partner Link fault.

In case of a Partner Link fault, the fault policy is invoked and based on the fault the actions defined in the fault policy are executed. The default fault policy will have the" 'ora-java' action which will call the error logging and notification functions. After the 'ora-java' action returns, a 'human intervention' action is executed that shows the instance as 'waiting for input' in the BPEL console.

In case of a non-Partner link fault, the fault gets propagated to the catch all block of the BPEL process. Within the catch all block the AIA fault is constructed and subsequently the 'AIAAsyncBPELErrorHandling' process is invoked. This process is responsible for doing the error logging and notification. The next activity in the catch all block will either 'reply with the fault' (in case the faulted process was invoked synchronously from another BPEL service) or re-throw the fault.

When a BPEL Service is calling another BPEL service synchronously even the non-Partner Link errors in the invoked service show up as Partner Link error for the calling service.

The error handling approach defined above could be used for the following BPEL Process scenarios

1. Service 1 invokes Service 2 synchronously (Service 2 is like a sub-process)
2. Service 1 invokes Service 2 asynchronously (Service 2 is like a sub-process)
3. Standalone Service 2

Table 1.13: Error Handling scenarios

Scenario	Fault Type	Fault Handling
Service 1 invokes Service 2 synchronously	Non-Partner Link Fault occurs in Service 2	1. The Fault will be caught by the catch all block. 2. Service 2 will reply with a Fault back to Service 1. 3. Service one receives a Partner Link Fault 4. Default Fault Policy is invoked. a. The first action executed for the fault will be the "ora-java action". The fault will get logged and an error notification is sent based on the roles setup in the Worklist configuration. b. The Java action returns a value. For all values that are returned we will call the human intervention action so that the fault is now available in the BPEL console. 5. Since we have called the human intervention as the last action, the BPEL instance will appear as waiting for input in the BPEL console. In the console the user can a. Modify/correct the payload and

		resume/restart the erred BPEL service instance b. Resume/restart the erred BPEL service instance without modifying the payload c. Terminate the erred BPEL service instance
Service 1 invokes Service 2 synchronously Service 1 invokes Service 2 asynchronously Standalone Service 2	Partner Link Fault occurs in Service 2	1. Default Fault Policy is invoked. a. The first action executed for the fault will be the “ora-java action”. The fault will get logged and an error notification is sent based on the roles setup in the Worklist configuration. b. The Java action returns a value. For all values that are returned we will call the human intervention action so that the fault is now available in the BPEL console. 2. Since we have called the human intervention as the last action, the BPEL instance will appear as waiting for input in the BPEL console. In the console the user can a. Modify/correct the payload and resume/restart the erred BPEL service instance b. Resume/restart the erred BPEL service instance without modifying the p ayload c. Terminate the erred BPEL service instance
Service 1 invokes Service 2 asynchronously Standalone Service 2	Non-Partner Link Fault occurs in Service 2	1. The fault will be caught by a catch all block 2. The error information will be mapped to the AIA fau lt which is the input the AsyncErrorHandling BPEL process. 3. The AsyncErrorHandling BPEL process will be invoked to do the error notification and error logging. 4. The fault will be re-thrown so that the instance appears as faulted in the BPEL Console

3.4.6 Message Ordering

Messaging order is a requirement in any message-oriented integration which contains dependencies between message actions or even between data interfaces.

Part of the functional design was to identify what message ordering requirements are on the data interfaces and how to solve them. It has been decided that the message’s target applications will deal with any ordering issues for the identified dependencies and that the middleware will not cater for message reordering or order related error handli ng.

Table 1.14: Message order functional rational

Data Interface	Message Types	Reason

Vendors	Creation Address modification	No message dependencies between vendor creations; For vendor modifications without a previous creation received, RIB hospital will deal with ordering errors;
Items	Creation Header modification Supplier creation Supplier country modification Supplier country dimensions modification UPC creation UPC modification UPC deletion	No message dependencies between item creations, item supplier creations and item UPC creations ; Unlikely to have any kind of modifications on the same transaction and if it occurs without a previous creation received RIB hospital will deal with the ordering errors; For UPC Mod and Deletions, RIB hospital will deal with ordering errors;
Purchase Orders	Creation Header modification	No message dependencies between PO creations; PO modifications will not occur on the same messages transaction.
Stock Orders	Creation Header modification Detail deletion	No message dependencies between creations; For transfer modifications and deletions without a previous creation received, RIB hospital will deal with ordering errors;
Stock Order Status	Creation	No message dependencies between creations.
Inventory Adjustments	Creation	No message dependencies between creations.
RTV	Creation	No message dependencies between creations.
Receipts	Creation Modification	No message dependencies between creations; Message dependencies between receipt modifications and receipt adjustment modifications will be dealt in the Mainframe processes; Modifications for the same receipt, knowing that Mainframe closes the PO when processing a receipt and assuming that the receipts will be sent from ORWMS only when the appointment is closed for the PO, message ordering on this scenario is not an issue.

3.5 Conclusions

The technology decisions reviewed on this chapter have impacts on the following project phases, as technical design, development and even the physical architecture, since the functional applications determine how the physical infrastructure should be deployed and what should it cater for.

These constrains are reflected on the next chapter, which describes the project from a technical point of view, including the development and migration phases.

Chapter 4

Implementation

This chapter describes the solution for this project at the technical level, including the technical components design, the development and deployment methodology used, the physical architecture and a overview on the post-development steps, as initial data load (data migration) for ORWMS and testing phases.

4.1 Technical Design

4.1.1 Enterprise Business Services

The Enterprise Business Services for this integration implementation will be Oracle Advanced Queues that hold messages published by the requestor ABCS to be consumed by the provider ABCS.

This type of EBS has been chosen since there is no transformation or routing requirements but it is necessary to keep this implementation extensible.

The following queues are implemented.

Table 1.15: EBS queues

QUEUE	Publisher	Consumer
AIA_VendorJMSQUEUE	SyncVendorMainframeReqABCSImpl	SyncProductMainframeProvABCSImpl
AIA_ProductJMSQUEUE	SyncProductMainframeReqABCSImpl	SyncVendorMainframeProvABCSImpl
AIA_PurchaseOrderJMSQUEUE	SyncPOMainframeReqABCSImpl	SyncPOMainframeProvABCSImpl
AIA_TransferJMSQUEUE	SyncTransferMainframeReqABCSImpl	SyncTransferMainframeProvABCSImpl
AIA_SOStatusJMSQUEUE	SyncSOStatusRIBReqABCSImpl	SyncSOStatusRIBProvABCSImpl

AIA_InvAdjustJMSQUEUE	SyncInvAdjustRIBReqABCSImpl	SyncInvAdjustRIBProvABCSImpl
AIA_RTVJMSQUEUE	SyncRTVRIBReqABCSImpl	SyncRTVRIBProvABCSImpl
AIA_ReceiptJMSQUEUE	SyncReceiptRIBReqABCSImpl	SyncReceiptRIBProvABCSImpl

4.1.2 Receiving Processes

Each ABCS implementation has a BPEL process that's responsible for the connectivity with the queues or topics and to call the transforming, routing and publishing BPEL process. This is required for the error handling framework, in order to transform a partner link error on the child process into a non-partner link process. The communication between these two processes is synchronized.

The following diagram is a general representation of the BPEL processes responsible for this pre process.

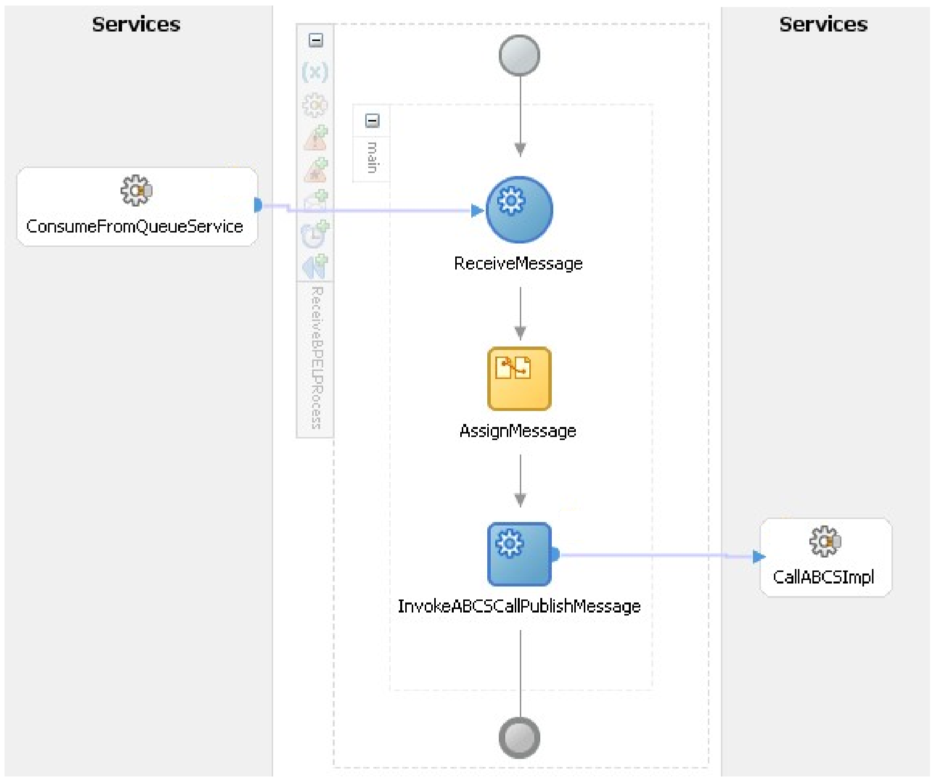


Figure 1.25: BPEL skeleton for a receiving process

4.1.3 Mainframe Requestors

Mainframe requestors consume fixed length messages from the Mainframe queues and published them in the middle queues, transforming them into RBMs.

The following diagram is a general representation of the BPEL processes for this type of ABCS.

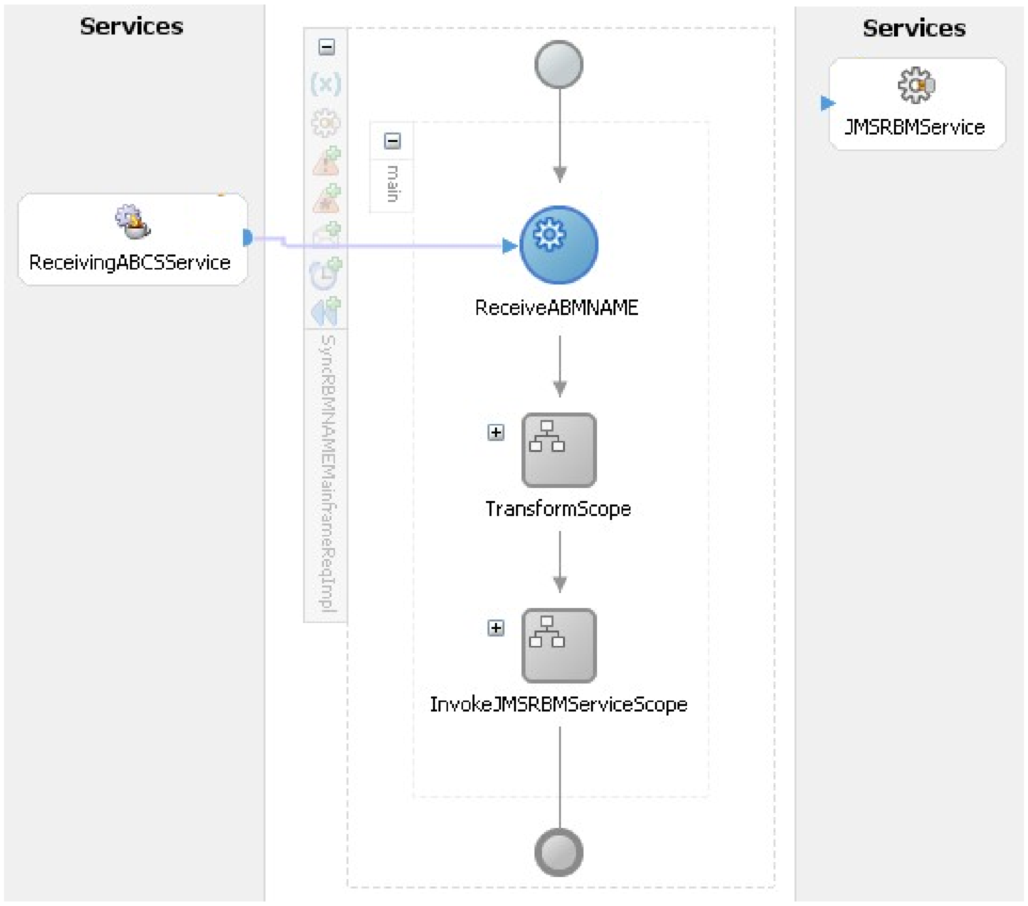


Figure 1.26:BPEL skeleton for Mainframe requestors

The receiving process for this BPEL process receives the messages from a MQ adapter which is responsible to consume messages from the Mainframe MQ Series queues.

A receive activity assigns a variable with the messages received and then the transformation activity(ies) transform(s) the fixed length ABMs into RBMs. For vendors and items, more that one transform activity is required since the target XSD depends on the message type received.

Invoke activities (different invoke activities are required for each transformation activity) call the JMS publishing adapter for the EBS queues, publishing the EBMs in the respective queues.

The triggering event for this type of ABCS is a message on the MQ queues that is received by the Oracle MQ adapter.

4.1.4 Mainframe Providers

A mainframe provider transforms RBMs consumed from the middle queues into RIBMessages and publishes the transformed message son the ORIB JMS topics.

The following diagram is a general representation of the BPEL processes for this type of ABCS.

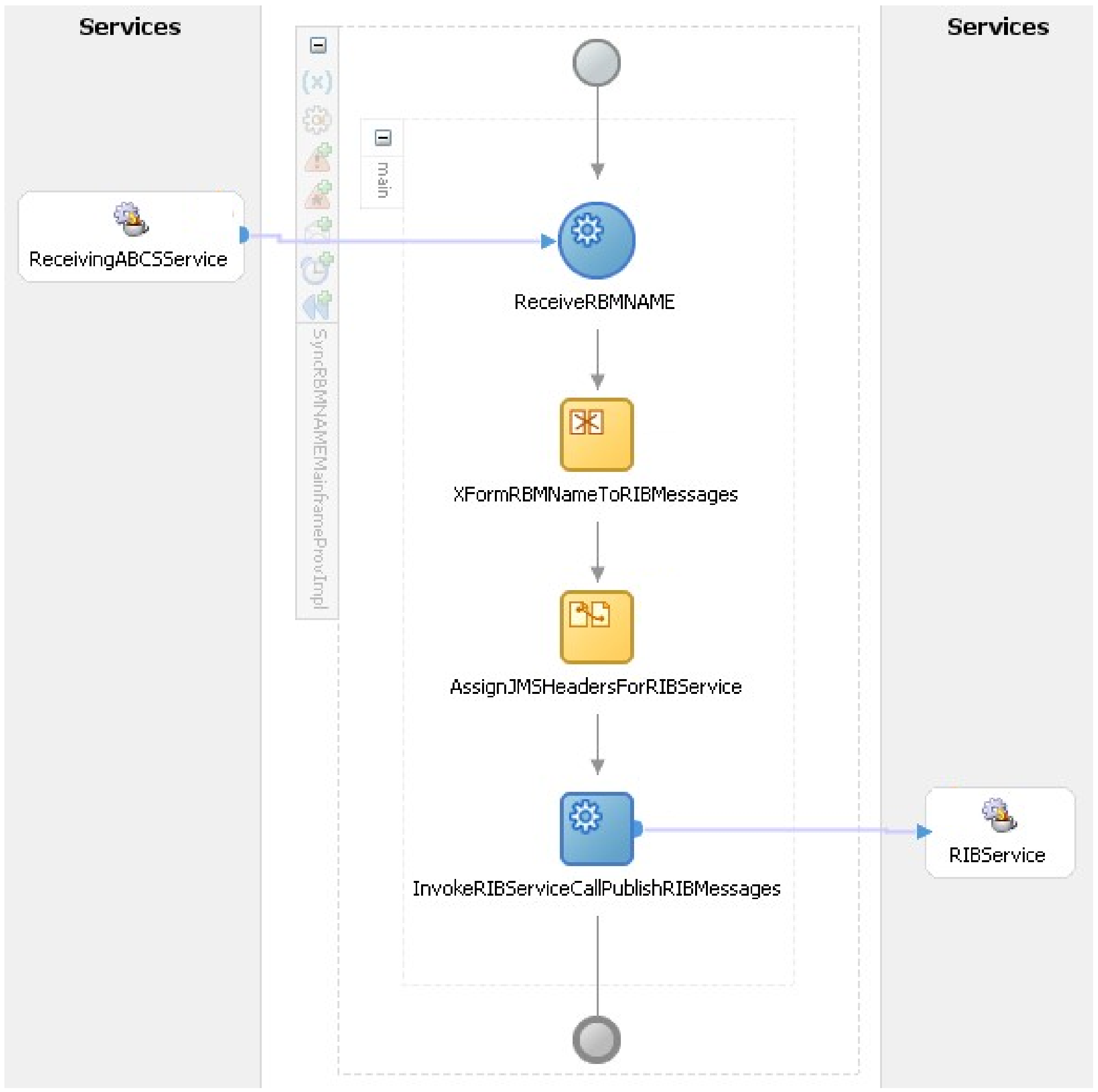

Services
Services
ReceivingABCSService
main
SyncRBMNAMEMainframeProvImpl
ReceiveRBMNAME
XFormRBMNameToRIBMessages
AssignJMSHeadersForRIBService
InvokeRIBServiceCallPublishRIBMessages
RIBService

Figure 1.27:BPEL Skeleton for Mainframe providers

The receiving process for this BPEL process receives the messages from the messages from a JMS adapter responsible to consume messages from the RBM topics;

A receive activity assigns a variable with the messages received and then the transformation activity transforms the RBMs into RIBMessages ABMs, it is a transformation that has different source XSDs but the same target XSD;

An assign activity assigns JMS header properties required for the RIB JMS interface using a variable from the JMS outbound header partner link;

An invoke activity calls the JMS publishing adapter for RIB JMS, publishing in the respective topics.

The triggering event for this type of ABCS is a message on the middle AQ queues that is received by the Oracle JMS adapter.

4.1.5 ORIB Requestors

These ABCS get the ORIB ABMs (RIBMessages) from the ORIB JMS topics and transform them into RBMs to be published on the middle queues.

The following diagram is a general representation of the BPEL processes for this type of ABCS.

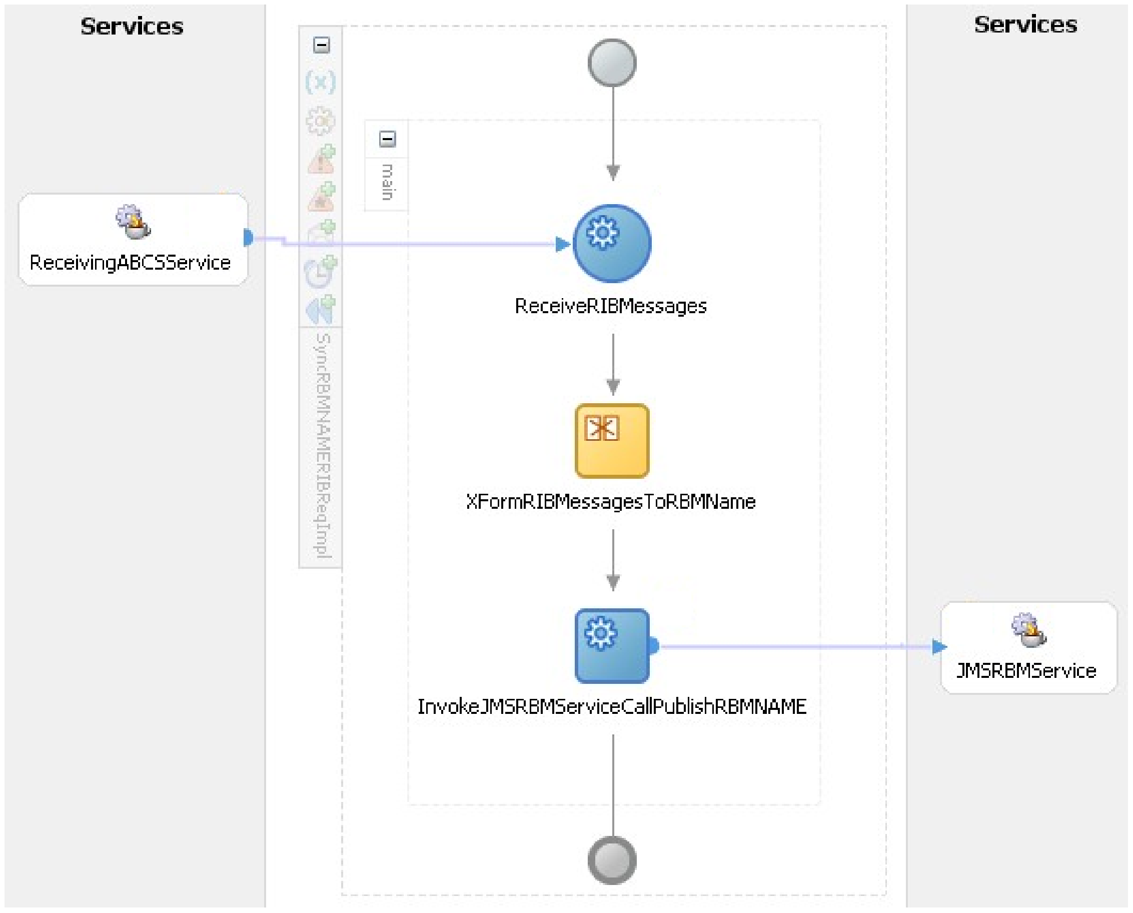


Figure 1.28:BPEL skeleton for ORIB requestors

The receiving process for this BPEL process receives the messages from a JMS adapter responsible to consume messages from RIB topics.

A receive activity assigns a variable with the messages received and then the transformation activity transform the RIB messages into RBMs, this transformation activity has the same source XSD but different target XSDs;

Invoke activities call the JMS publishing adapter for the EBS queues, publishing the EBMs in the respective queues.

The triggering event for this type of ABCS is a message on ORIB topics that is received by the Oracle JMS adapter.

4.1.6 ORIB Providers

These ABCS get the RBMs from the middle queues and transform them into fixed length messages to be published on the Mainframe queues.

The following diagram is a general representation of the BPEL processes for this type of ABCS.

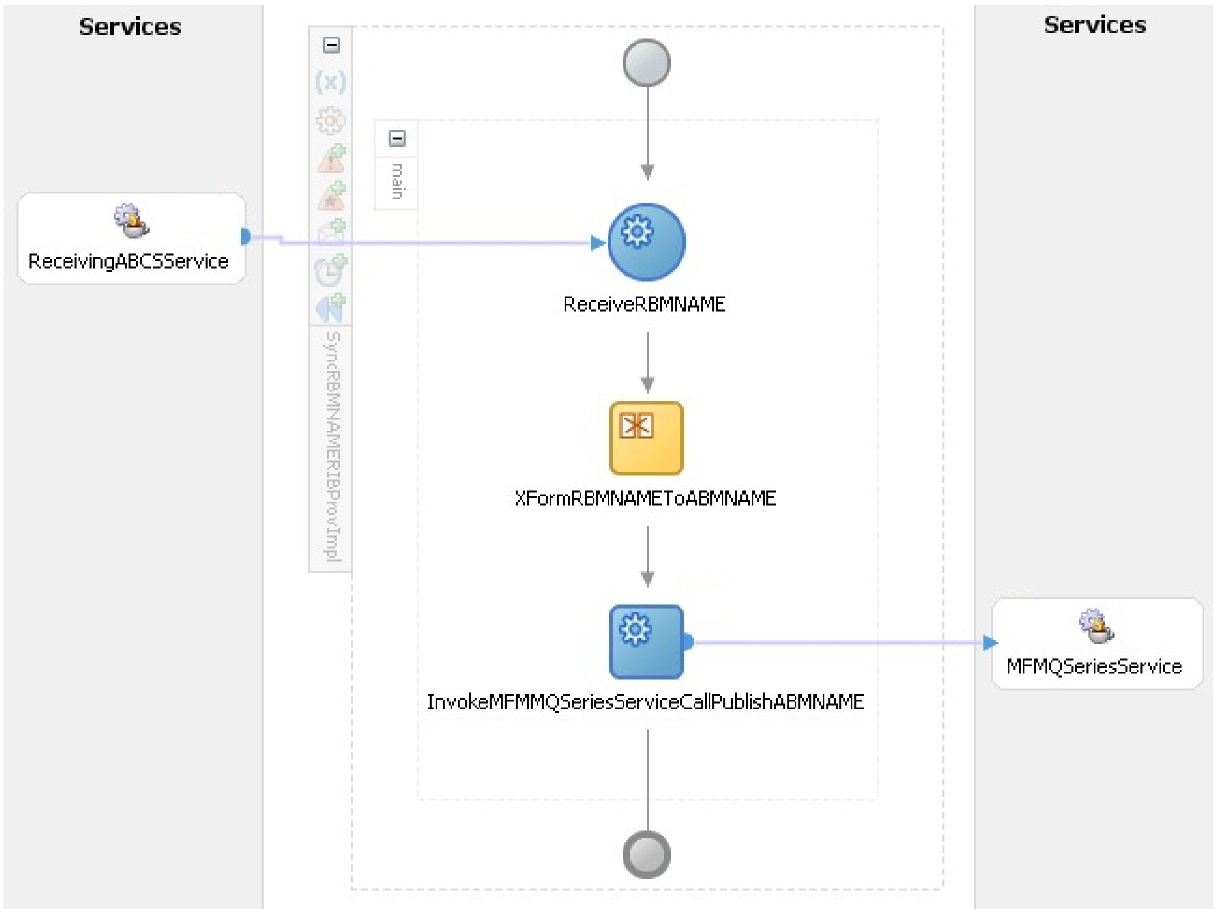


Figure 1.29:BPEL skeleton for ORIB providers

The receiving process for this BPEL process receives the messages from a JMS adapter responsible to consu me messages from the RBM topics.

A receive activity assigns a variable with the messages received and then the transformation activity transforms the RBMs into Mainframe fixed length ABMs;

An invoke activity calls the MQ publishing adapter for Mainframe MQ, publishing in the respective queues.

The triggering event for this type of ABCS is a message on the middle AQ queues that is received by the Oracle JMS adapter.

4.2 Code Deployment Methodology

4.2.1 Organisation, Roles and Responsibilities

The following stokeholds are part of the code deployment methodology:

- The Technical Lead is responsible for the Release Management procedures;
- All technical team members are responsible for adhering to the procedures set out;
- The Release Manager is responsible for delivery of a release to the Deployment Team;
- The Deployment Team is responsible for the installation of a release or patch.

4.2.2 Construct Release

The procedure for a deployment construction is as followe d:

- Code is released to the Deployment team as a Release Pack. This pack is a zip file that contains the interface module code that comprises of BPEL and ESB modules (e.g. ProductImpl);
- Release numbers are sequential;
- A build is promoted from the Subversion repository to release version into a separate area in the Configuration Management (CM) repository that stores code that has been or is about to be released.

4.3 Physical Architecture

This integration architecture is required to support more than one (frozen) warehouse ORWMS instance and the constraint of having the ORWMS server located on the warehouse, therefore the following architecture was conceived in order to accommodate these requirements using ORIB distributed functionalities.

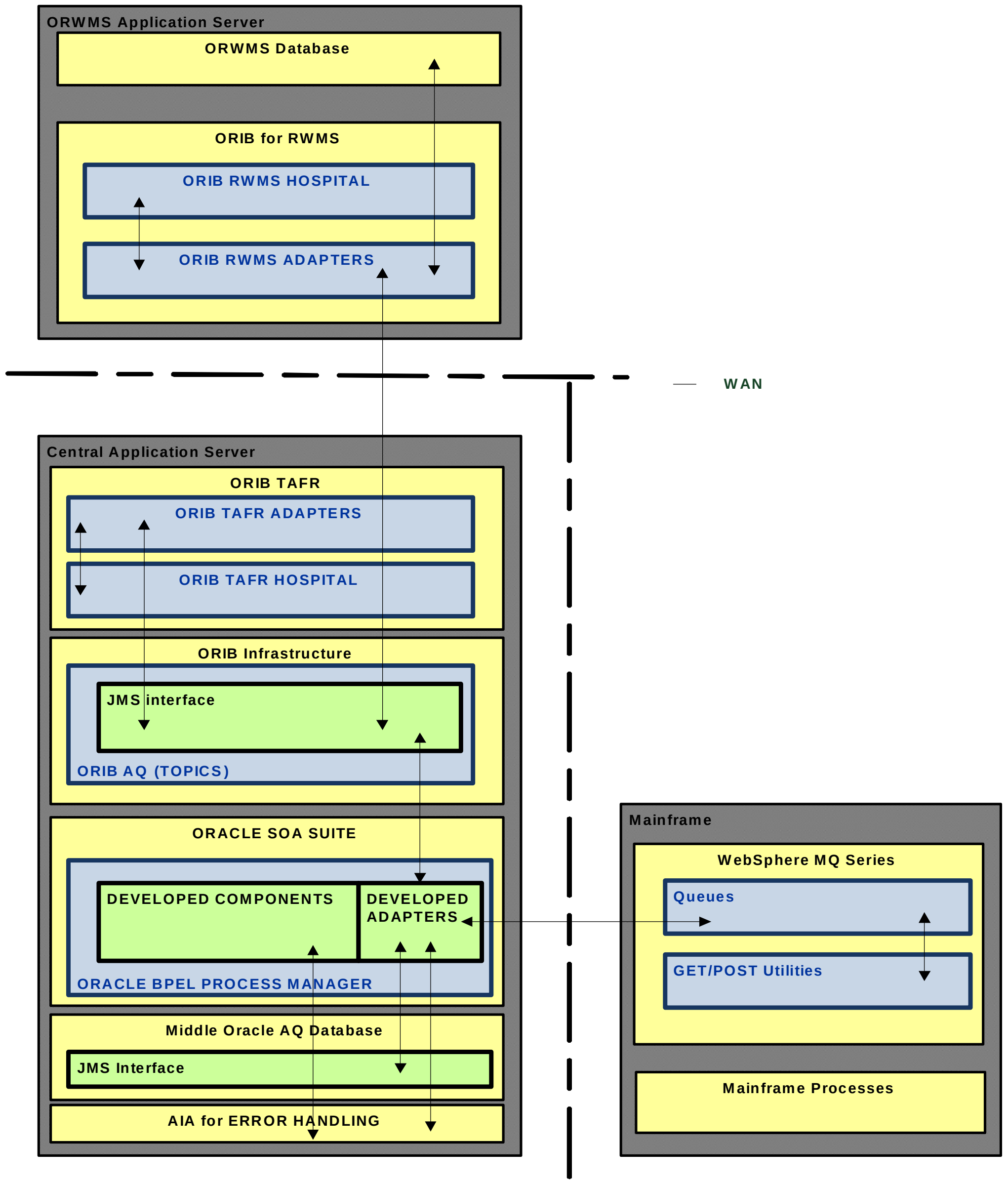


Figure 1.30:Application servers architecture

Besides the application server logical constraints and requirements, the client's infrastructure team required disaster recovery on the physical servers for all the Oracle products, which means that this project infrastructure should be built on Oracle Real Application Clusters (RAC)

[ORAC] on HP 9000 Superdome [SPDM] servers having separate servers for applications and databases.

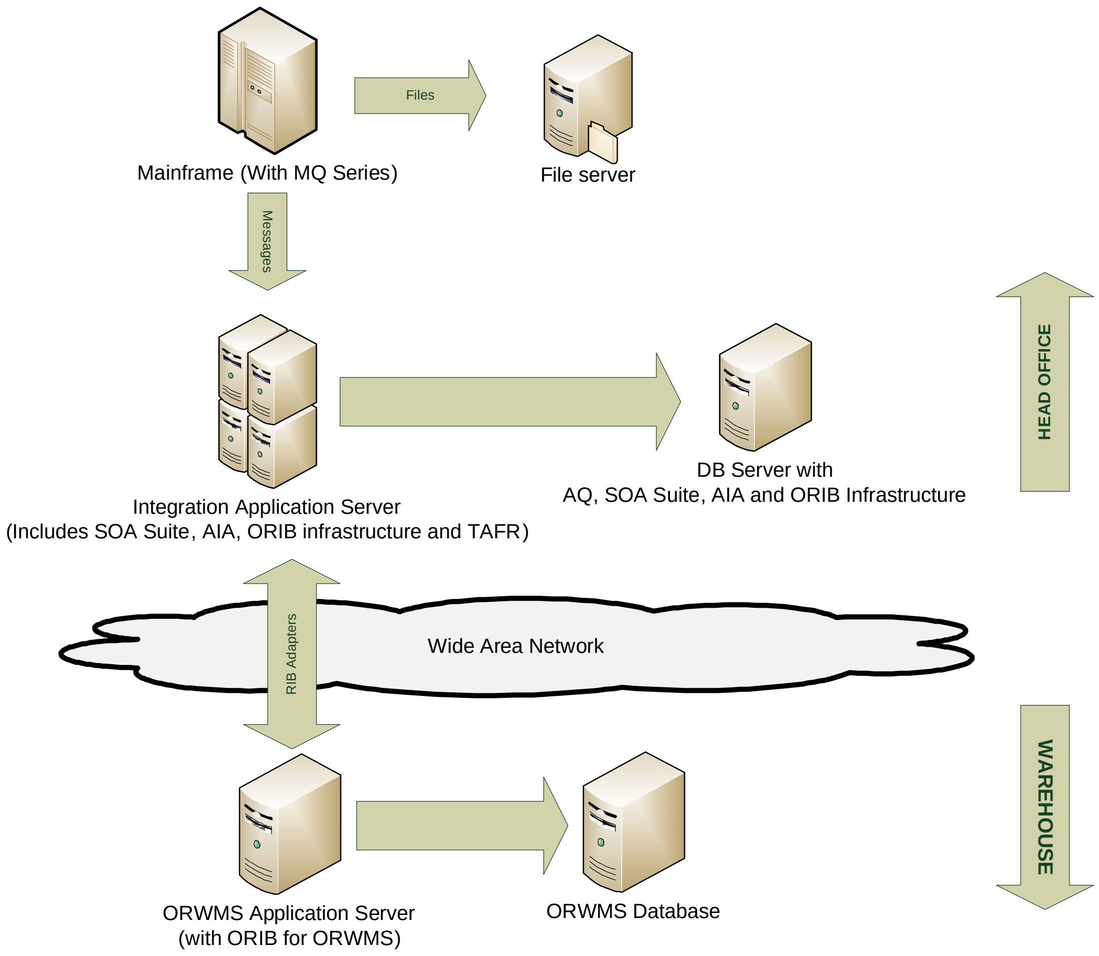

Files
Mainframe (With MQ Series)
File server
Messages
HEAD OFFICE
Integration Application Server
(Includes SOA Suite, AIA, ORIB infrastructure and TAFR)
DB Server with
AQ, SOA Suite, AIA and ORIB Infrastructure
RIB Adapters
Wide Area Network
WAREHOUSE
ORWMS Application Server
(with ORIB for ORWMS)
ORWMS Database

Figure 1.31:Physical servers architecture

4.4 Development

The development phase, which includes two cycles of internal unit tests, was performed in India, by the Wipro Technologies offshore team (two resources) allocated for this project and other retail integration projects for the same client.

The support, development plan and decision making was part of the author's responsibilities, along with the delivery of the code.

The communication between the author and the development team was made on a daily basis between phone calls and emails which is considered to be a very close work between the two entities.

All the components and objects were developed according to the Oracle AIA naming conventions for standardization purposes.

The biggest issues found during this phase were the time difference (plus 5 hours than UK) that required a different daily work plan than a normal start of the day meeting and the communication between the client and the development team which required a constant two side update also on a daily basis. Despite these issues, the development phase of this project started and ended as planned and with the expected delivered.

4.5 Data Migration

The migration of the initial (foundation and transactional) data from Mainframe to ORWMS along with the data analysis is part of the data migration team (Wipro Technologies) scope for this project. Since, like the integration team, the data migration team supports the business team (logistics), the last one decided that data migration should be done trough the developed interfaces for data integration. The following data will be migrated.

Table 1.16: Data migration types

Data	Type	Load Type	Developed by
Locations	Foundation	Automatic and Manual	Data Migration
Supplier	Foundation	Automatic	Integration
Items	Foundation	Automatic	Integration
Locators	Operational	Automatic	Data Migration
Users	Operational	Automatic and Manual	Data Migration
Inventory Stock-Bulk	Operational	Automatic	Data Migration

Despite having the integration components to get the data from the Mainframe, there was still an issue on how the data push should be done, since the Mainframe processes designed for integration were made on an ad-hoc basis, getting the data from modification tracking files, which is not valid for a full data gathering and load.

This issue was resolved by getting the data from a data warehouse relational database and new processes development done by the client technical team, in order to forward all the data to be migrated into the MQ queues and format it according to the fixed length message definition.

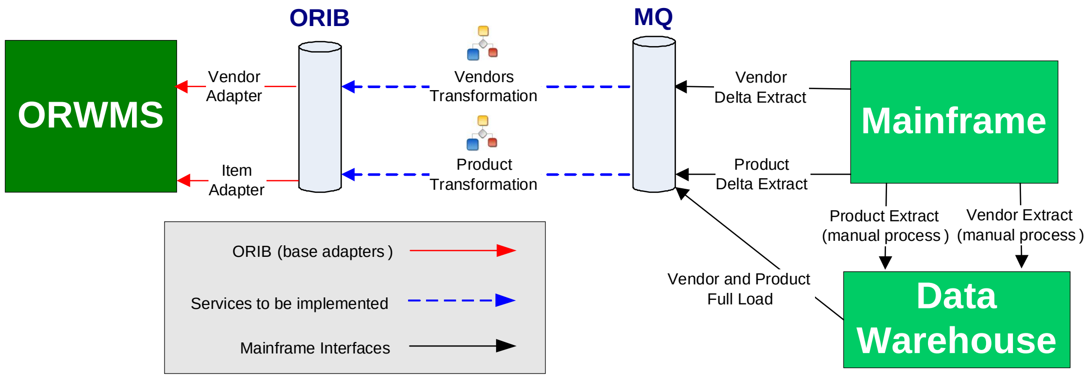


Figure 1.32: Vendor and Product Data Migration components

4.5.1 Data Analysis

Every data migration requires analysis of the source data and how it maps to the target data.

In this special scenario, for logistics data migration of products (or items) and vendors using the integration components for ORWMS load, the effort of the data migration team was based on the analysis of the source data which required rationalization (mapping from Mainframe logistics concepts and data definition into ORWMS concepts and data definition) and data cleansing (removal or filter of data from the source application in order to prevent data not accepted by the target application).

This work was completely a responsibility of the data migration team but the author gave support from an integration point of view (for example, which data will be enriched on the integration components therefore not requiring Mainframe load) and from an ORWMS point of view (for example, which types of item details are required when creating an item on ORWMS).

4.6 Testing

The following sections covers all the different types of test phases planned to be executed for the logistics project testing which includes the data integration components except for the test phases after performance tests which are not documented, since the author’s support on those phases is only relevant if there are significant modifications to the initial business requirements and acceptance criteria. It also covers the entry, exit criteria and the responsibilities of each test execution.

4.6.1 Unit Testing

The objective of unit testing is to ensure that the developed objects function according to their functional and technical specifications. It is a basic level of code and configuration testing.

Entry criteria:

- Approved requirements definition document;
- Approved functional specification document;
- Approved technical specification document;
- Identified all the component testing scenarios and test cases;
- Development environment and test data ready for unit testing;

Exit criteria:

- 100% of all Priority 1 and 2 level defects tested and fixed;
- Successful execution of all unit test cases in development instance;
- Code review completed;
- Logging of the defects found during execution;
- Testing evidence reviewed and approved by testing team and quality function;
- Quality gate document sign off.

Integration team responsibilities:

- Prepare and execute test scenarios for the integration;
- Identify data requirements for component test scenarios;
- Raise and re-test component testing defects;
- Record test results and capture test evidence;
- Raise and progress component testing issues and risks;
- Sign-off at the quality gateway meeting to indicate testing completed to their satisfaction.

Testing team responsibilities:

- Manage and co-ordinate with the integration team in component testing activities;
- Provide testing related training to enable the development team to conduct component testing (e.g. Quality Centre training for test planning and defect logging and tracking);
- Support and help in raising defects;
- Uploading the results, defects, test scenarios and test cases of Component testing to quality centre.

4.6.2 System Integration Testing

The objective of integration testing for the logistics project is to test interfaces of ORWMS with Mainframe.

Entry criteria:

- ORWMS configuration is complete and has been component and functional tested;
- All known issues have been captured and all priority one & priority two level defects of all previous test phases have been tested and fixed;
- All test cases are loaded into quality centre;
- Test environment ready with the integration layer components;

Exit criteria:

- 100% of integration test cases have been executed & results documented;
- For failed test scenarios, no priority one or priority two level defects are outstanding, or alternatively, a solution identified that can fit in next cycle of integration testing is documented;
- List of outstanding defects, if any, are published;
- Quality gate meeting with test phase sign off.

Responsibility:

- Logistics testing team.

Support required:

- From Mainframe applications team;
- From integration development team;
- From logistics development team.

4.6.3 Performance Testing

The objective of performance testing is to validate the system performance against the defined performance Service Level Agreement (SLA) as mentioned in the non functional documentation for speed requirements, scalability (number of users and transaction volume) and reliability.

Entry criteria:

- Non functional requirements documentation signed off;
- System integration test phase completed;
- Performance test preparation documentation have been completed and reviewed;
- Availability of dedicated performance test environment;
- Test environments that will interface with the to be tested systems are available and connectivity is established;
- Base volume data for the test has been prepared and validated by data migration team;
- Application build is complete with all the system and application level configuration setting;
- Load balancing rules are implemented;
- Necessary Security Socket Layer (SSL) certificates are enabled;
- Agreed support requirements from various project teams prior to and during performance test execution including analysis of the log files and results;
- Batch components are stopped on the production test environment to ensure that there is no process load on the participating environments..

Exit criteria:

- All business scenarios successfully are executed, no outstanding performance defects at severity one or two, with acceptable number of lower severity defects;
- Any performance issue or bottleneck identified during the test execution is fixed and retested;
- Any unresolved issue that will be carried over into the production environment is documented and signed-off by the client;
- All environment modifications and fixes that are put in place to resolve any performance related issues are captured and documented for future reference;

- Breakpoints are acceptable to the business (i.e. the user volumes at which system performance deteriorates to unacceptable levels exceeds business requirements);
- Test completion report produced and signed off by the stakeholders.

Responsibility:

- Logistics testing team.

Support required:

- From Mainframe applications team;
- From integration development team;
- From logistics development team.

Chapter 5

Conclusions and next steps

This final chapter of this document presents the objectives for this project and how they were achieved according to the evaluated results. Also presents some extra work related to this project done by the author and it ends with the author's next steps on the project and some recommendations for future related work as well as a descriptive summary of the lessons learned by the author when working in a long programme of IT projects.

5.1 Objectives Completion

The following text fragment is part of the project proposal and specifies the project objectives.

"Analysis, design and development of an integration solution between a legacy system and a warehouse management system, for a retail client.

Integration between a legacy system and Oracle Retail Warehouse Management System with initial verification of the Oracle SOA Suite (ESB+BPEL) functionalities, Oracle Retail Integration Bus and Message-Driven Beans on a Oracle Application Server environment for a retail client based on the United Kingdom. The developed solution will initially serve a frozen products warehouse, where the message load is not critical, providing an ad-hoc implementation of the integration with the legacy system. The warehouse management system implementation will be standalone, not having other Oracle Retail suite products. The integration will have as touch points fixed length files on the legacy side and Java Messaging Services on the other side, taking advantage of the integration application of the Oracle Retail suite that implements the referred service.

The project will start with a proof of concept to verify the viability of the several possible approaches and ensure the next phases' decisions from an implementation point of view. Then the project will continue with the flow and relative mappings analysis in order to meet the final retailer needs, followed by the functional and technical design phases ending with the development, unit and integration testing of the proposed solution. The aspects related with the advantages of using bus-oriented integration architecture during all the project phases should also be analysed, in order to maintain consistency with the Oracle Retail suite integration bus." [PPD]

The following table describes the completion of the objectives that were defined for this project.

Table 1.17: Objectives completion

Objective	Achievements
Integration scope	All objectives were achieved except for the legacy file integration due to legacy's constraints. This issue was resolved recurring to a WebSphere MQ module for the legacy system, using queues as the touch point.
Proof of concept	All objectives were achieved except for the legacy application touch point connection. A connectivity test was made afterwards using the WebSphere MQ module which was successful.
Flows and mappings analysis	All objectives were achieved but the outcome of this analysis can only be identified when the final solution is implemented and live.
Functional design	All objectives were achieved: the functional design document was approved by the client.
Technical design	All objectives were achieved: the technical design document was approved by the client.
Development	The development phase was supported by the author and the developments were completed as specified.
Unit testing	The developed components passed this phase without any bug or defects. All components passed all the unit test cases.
Component integration testing	This project phase was ongoing at the time this report was written.
Analysis of the bus-oriented architecture	Objective completed. This report was done with the help of that analysis.

5.2 Related Side Work

This integration project on this client included one more data interface that wasn't on the scope of the presented architecture.

This interface serves to get a comparison report between the stock registered on ORWMS and the stock on hand expected on Mainframe. If stock levels are not equal then the result will be either store orders which cannot be satisfied, and as a consequence a high nil pick rate in the distribution centre or there could be outstanding stock in the warehouse which cannot be picked.

5.2.1 Functional Architecture

In order to get a stock on hand comparison report on Mainframe, it was necessary to build a preparation batch program that writes a file with the stock available on ORWMS and then it is transferred into the Mainframe server to be compared with the stock on hand expected snapshot from the mainframe database.

The technology chosen to transfer this file was the same as for the other data interfaces, in order to take advantage of the error handling mechanism, quick development functionalities and a proven Mainframe touch point.

The following diagram represents all the components involved in this data integration.

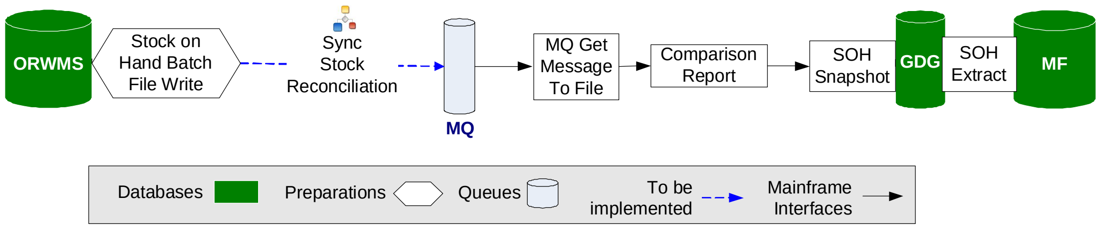


Figure 1.33:Stock reconciliation functional architecture

The design responsibilities are as follows:

- ORWMS support: Wipro Logistics team;
- Batch file write: client;
- Sync stock reconciliation BPEL process: author and Wipro integration team;
- MQ, MQ get message to file utility, comparison report, stock on hand snapshot, etc: client.

5.2.2 Component description

Since the messages for this component should be treated as opaque, message definition and documentation is not required.

The messages subscribed from the source application should be published on a MQ queue, on Mainframe.

This interface does not have to comply with the defined functional architecture as it does not requires extensibility, routing, reusability or any kind of transformation between the source and the target application.

5.3 Next Steps

The next phases of this project where the author is involved are the several testing and deployment phases.

Since the integration team purpose is to support the business requirements, the author's objectives for these phases are on the support and maintenance scope.

When an integration is specified with the characteristics of the one presented on this document is required for any type of client, small or big, with high or low skilled IT resources, the author recommends that Oracle AIA foundation pack is implemented, with the required customizations and developed components for the application integrations requested. This architecture framework will have more process integration packs in the future which means that more Oracle applications will have out of the box integration components and, for the ones that are not available from this software house, AIA, with Oracle SOA suite, can always provide a standardized service and message oriented middleware integration solution framework and architecture.

5.4 Lessons Learned

When being a part in a large system implementation as operative member of a system integrator and consulting company the feeling of responsibility for its size, length and impact can be sometimes overwhelming but, after some weeks, the mind should be set in the right direction and should know that there will be some nice times but much more not so nice time since, after all, this is client relationship management on a long run. The key for this marathon is to enjoy the success of the victories in the middle.

Typically, big retailers have complex architectures and even, sometimes, specific architectures for different purposes or business areas. This usually means that they have different applications, user interfaces, integration interfaces and technologies and, when initiating a project of this size, is critical to have the as-is architecture which can be hard to obtain or even not helpful enough due to the lack of rationalized data (data with different keys or with different structures) and this is where normally a concept as data rationalization is brought to the client.

Another issue when implementing big systems architectures on a client with some history is the age of the existing applications as usually only the subject matter experts (SMEs) (which is possible that doesn't exist for some applications) know what the programs do, what the source code means (if it even exists), especially if there is no readable documentation to the common systems analyst.

Since this project was integration of a part of a new system with the main legacy application, all of these issues came along and one can struggle if it's not prepared. Another issue on this type of projects is when to stop a legacy application. Questions like "The new applications support all this functionalities?" and, for integration, "Are all interfaces to data migration identified?" or even if the legacy applications can't be shutdown, to know which applications will be the master for each business process and what are the data interfaces required or if its possible to adapt the legacy to work with the to-be architecture can define the success or failure of a systems implementation.

Many other issues can and probably will in this type of projects. Issues like the number (and type) of legacy environments, size of the client's team and alignment, the morale of the client, the decision making process, the technical and business oriented vocabulary discrepancies between the client and the systems integrator will make the definition of scope of the programme a very time and resource consuming task.

The methodology to approach all these issues is based on client awareness and communication on early stages of the programme. Jobs like dependency awareness, as-is architecture documentation, identify major integration issues, having the team prepared on site are our of the systems integrator team hands.

Then, it all starts on the scope acknowledgment, getting to know the business and functional goals, decisions and requirements, understand what will remain (and for how long) in legacy applications and, at the operational level, to know what interfaces will be exchanged between the new and the legacy applications (most of it decided by the legacy SMEs, who know what are the legacy needs).

Despite knowing that is necessary to mind the legacy applications, it is also important to never forget why the programme exists: the new applications or suite. Starting from the new applications and discuss with the client what will be integrated and how (from applications and data interfaces and, if possible, even to data attributes) is another starting point as relevant as the ones presented before.

Since it's obvious that the success key is communication, the systems integrator must give the example, with a good team structure with good communication, with offshore planning (having in mind that most of the development and technical work on these types of programmes is done on the systems integrator offices), investing on team building, either inside and outside the team (with open workshops about functional concepts or even provide documentation of the new system to everyone for example) and to know that behind a systems integrator team is a company with very skilled resources and they should be used when required.

Communication official processes (request, inform, discuss, etc) which are usually disregarded on small or medium processes are a must use on this type of projects since they exist to keep the right people on the information loop, at the right time and with the right escalation. It's also mandatory to follow or create other non communication processes like requirements sign off or review and to know when to adapt to the client's methodology and maybe losing some discussions to win the bigger one: a successful implementation.

Concluding, it's necessary, first of all, to respect the size of the programme, then to anticipate all the upcoming issues, to define responsibilities and boundaries, to know that communication is the key and, of course, to always be pragmatic and methodical.

References

[DLTR] "2008 Global Powers of Retailing" © 2008 Deloitte Development LLC, available at http://www.deloitte.com/dtt/article/0,1002,cid%253D185594,00.html

[EIR] Euromonitor International 2006 Report on Retail, available at http://www.euromonitor.com/

[IBMQ] IBM® WebSphere MQ 7, available at http://www-01.ibm.com/software/integration/wmq/v7/

[IVPP] Perrey R., Johnston A., Lycett M. and Paul R. (2004), "Value propositions: a new conceptualisation for integration" available at http://www.emeraldinsight.com/1741-0398.htm

[MOMMK] Message Oriented Middleware, Markku Korhonen, available at http://www.tml.tkk.fi/Opinnot/Tik-110.551/1997/mqs.htm#B2

[MSIPP] Integration Patterns, Patterns & Practices available online at http://msdn.microsoft.com/en-us/library/ms978729.aspx © Microsoft Corporation, June 2004

[OAIA1] Oracle® Application Integration Architecture – Foundation Pack 2.2.1: Concepts and Technologies Guide

[OBRS] Oracle® buys Retek software – DBA Oracle News 2005/04/25, available at http://www.dba-oracle.com/oracle_news/2005_4_25_buys_retek_software.htm

[OSAQ] Oracle® Streams Advanced Queuing User's Guide and Reference available at http://download.oracle.com/docs/cd/B14117_01/server.101/b10785/aq_intro.htm © Oracle Corporation, February 2009

[OAIA1] Oracle® Application Integration Architecture – Foundation Pack 2.2.1: Concepts and Technologies Guide © Oracle Corporation, September 2008

[OAIA2] Oracle® Application Integration Architecture – Foundation Pack 2.2.1: Integration Developer's Guide © Oracle Corporation, September 2008

[OFM] Oracle® Fusion Middleware portfolio presentation available at http://www.oracle.com/products/middleware/index.html © Oracle Corporation, February 2009

[ORAC] Oracle Real Application Clusters technical overview, available at http://www.oracle.com/technology/products/database/clustering/index.html© Oracle Corporation, May 2005

[ORMSa] Oracle® Retail Merchandising System Benefits, available at http://www.oracle.com/applications/retail/mom/merch_sys.html , © Oracle Corporation, , June 2008

[ORMSb] Oracle® Retail Merchandising System User Guide Release 13.0, © Oracle Corporation, June 2008

[ORPMIG] Oracle® Retail Price Management Installation Guide Release 13.0, available at http://download.oracle.com/docs/cd/E12440_01/rpm/pdf/130/rpm-130-ig.pdf, © Oracle Corporation, April 2008

[ORWMSa] Oracle® Retail Warehouse Management System User Guide Release 13.0, © Oracle Corporation, June 2008

[ORWMSa] Oracle® Retail Warehouse Management System Operations Guide Release 13.0, © Oracle Corporation, June 2008

[ORIBa] Oracle® Retail v.13 Documentation Library, Integration Bus Integration Guide available online at http://www.oracle.com/technology/documentation/oracle_retail.html © Oracle Corporation, June 2008

[ORIBb] Oracle® Retail v.13 Documentation Library, Integration Bus Implementation Guide available online at http://www.oracle.com/technology/documentation/oracle_retail.html © Oracle Corporation, June 2008

[ORWMSa] Oracle® Retail v.13 Documentation Library, Oracle Retail Warehouse Management System Operations Guide available online at http://www.oracle.com/technology/documentation/oracle_retail.html © Oracle Corporation, February 2009

[OSSSG] Oracle® SOA Suite Quick Start Guide 10g (10.1.3.1.0), available at http://download.oracle.com/docs/cd/B31017_01/core.1013/b28938/toc.htm © Oracle Corporation, 2006

[PPD] Project/Dissertation Proposal for the author, available at http://www.fe.up.pt

[SPDM] HP 9000 Superdome Server overview and features, available http://www.hp.com/products1/servers/scalableservers/superdome/ © 2008 Hewlett-Packard Development Company, L.P.

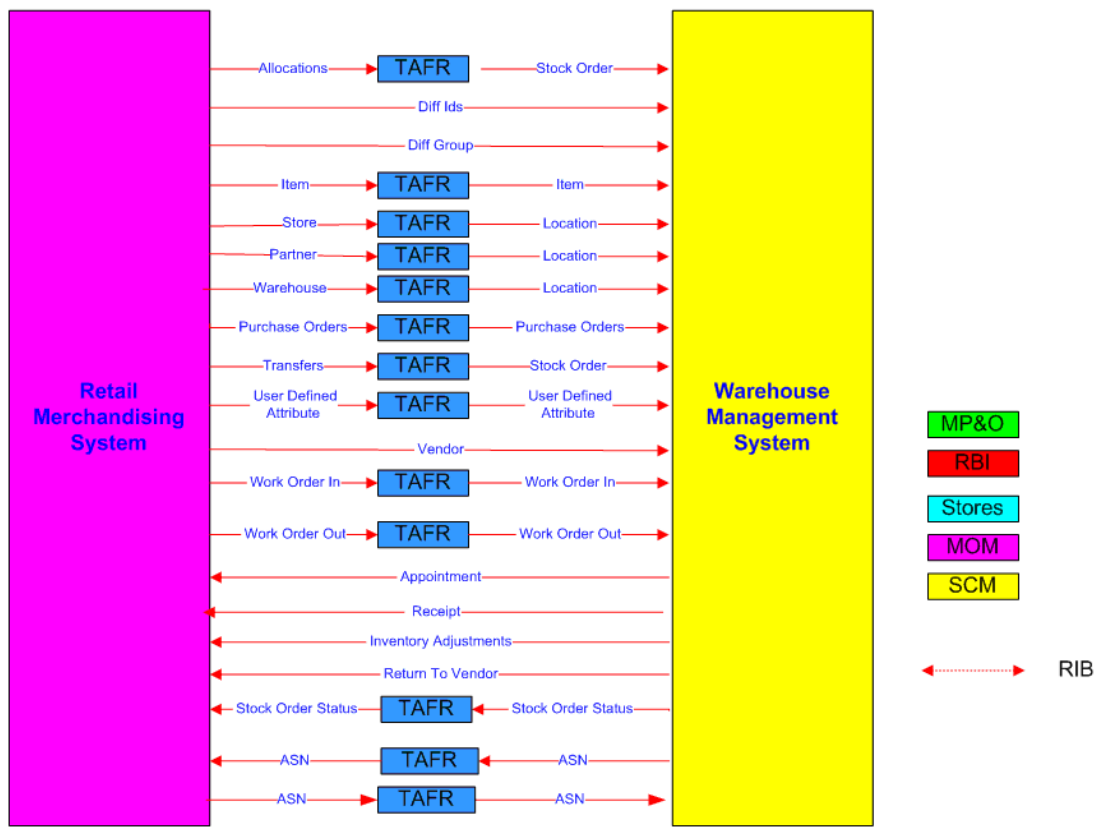
[WBPEL] Oasis® standard Web Services Business Process Execution Language definition, available at http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[XAJV] JAVA XA implementation specification, available at http://java.sun.com/j2ee/sdk_1.3/techdocs/api/javax/transaction/xa/XAResource.html

Appendix A

Integration interfaces between ORMS and ORWMS

This is a complete interface list between the Oracle Retail Merchandising System and Oracle Retail Warehouse Management System [ORIBa].



Disclaimer: This document is a general data mapping and reference guide for data coming in and out of Oracle Retail application systems via RIB/RSL messages; it is not meant to give detailed information about every possible data scenario.

Figure 1.34: Data mapping and reference guide for ORMS-ORWMS integration