ILP PROGRAM - ORACLE APPLICATIONS

Tata Consultancy Services

Interface/Conversion Study Guide

Author:	MD. Nazmul Hoda (TCS)
Creation Date:	Jan 7, 2015
Last Updated:	Jan 7, 2015
Document Ref:	ILP/ORACLEAPPS/INTERFACE/01
Version:	DRAFT 1A

Approvals:

<Approver 1>	Santanu Sarkar (TCS)
<Approver 2>	Shubho Chakraborty(TCS)

Document Control

Change Record

Date	Author	Version	Change Reference
07-May-15	MD. Nazmul Hoda	Draft 1a	No Previous Document

Reviewers

Name	Position

Distribution

Copy No.	Name	Location
1	Library Master	Project Library
2		Project Manager
3		
4		

Note To Holders:

If you receive an electronic copy of this document and print it out, please write your name on the equivalent of the cover page, for document control purposes.

If you receive a hard copy of this document, please write your name on the front cover, for document control purposes.

Contents

Table of Contents

How to use this manual

Video1: Script: Vid1-Introduction to the chapter and its content – Face recording.

This vide will introduce the material covered in this pdf, the goals,

1. How this document is organized
2. What is the purpose of this document
3. What will you achieve after going through the document and related videos
4. How to read this document
5. How does it relate to the work you will be doing on real project
6. Reference to other reading materials for further references

This manual has been organized as a step by step guide to teach how to create reports using Oracle Developer Suite 10G. The target audience is new comes to Oracle Developer suite. It assumes that the reader has basic knowledge of Oracle concepts and PL/SQL. After completing this course, you will be able to crate variety of reports using Oracle Developer Suite 10G.

This manual is organized to be read in a serial fashion and follow the instructions given in the document as it is. Practical examples are given in each section to guide you through every step. The tables referred here are common (shared) tables used by different batches, so care should be taken not to delete or update the rows which does not belong to you, this may create problem for the other batches. At the end of the course, you should delete the data you have created.

There are several symbols used to designate particular sections, which are described below:

- Describes the purpose of the section.

- Notes relevent to the scetion above

- This denotes the task to be completed by the audience on his own PC. The layout of the output has to be followed as it is. For any confusion, the faculty shoul dbe contacted.

1. Introduction

In this section you will learn what is meant by Customization, Interface, and Conversion

Oracle Application is a packaged software, which comes with a completed set of forms, reports, database objects and various other components to make a complete ERP. Oracle Application has been developed by taking Best Practices from various industries, and standards, which vary across the type of business and the country's legal and statutory regulations. For Example – The format of invoice will differ from country to country, again the taxation rules will be different in almost all countries of the world. So it is not possible for an ERP to cater for all variations.

Also, different business do business in different ways. For example, a company may operate 100% in cash, but another one offers a 35 days credit to the purchaser.

This kind of differences call for a change to be made in the Oracle Applications software which is called **Customization**.

Apart from this, an ERP system needs to talk to other systems. For example, the Banking transactions need to be fed to the Cash Management module of Oracle Applications, but the bank's IT system sits in the Bank's premises. So we need to develop a program which will pull the data from the bank's system and import the same inside Oracle Application. This is called an **Interface** program.

The term conversion comes into picture, when we build the ERP system for the first time. At this time, we need to import all necessary data which the organization has piled up so far, into the Oracle Applications. This data cannot be imported as it is. The data needs to be converted into the format, which Oracle Application ca understand and interpreter. In this case, we need to write a **Conversion** program, which perform the necessary conversion of data.

In the subsequent sections, we will go through details of these topics, along with examples.

2. Interface – Concepts and Need

\ in this section you will learn the concepts of interface

Before we indulge into the details, we need to understand few terminologies, which will be referred throughout this document:

Legacy System:

An organization normally runs a variety of software to support its functions. For example, Payroll System, Online Order Booking Portal, Document scanning software etc., and they may run on variety of systems like Windows, VMS, and DB2. When they implement Oracle Applications, some of them are replaced by Oracle Applications, but some are retained. Any such system which is not part of the Oracle Applications, is called a legacy system. Legacy system does not necessarily run on an old/outdated hardware or software.

Vendor:

A vendor basically means a person who supplies items to customers. Vendor is a more generic term and applies to anybody supplying goods and services. A vendor is rarely a manufacturer and gets products on consignment basis from the manufacturers. He can return unsold items and get his commission

Supplier:

A supplier is the one who makes the goods available. The supplier acts as the middleman between the retailer and the manufacturer. A supplier is also a distributor who makes the goods available to vendors. A supplier is many a times manufacturer also

The term Vendor and Supplier may be confusing at times, but for our purpose, both of them are the people or organization, who supply goods and services to us.

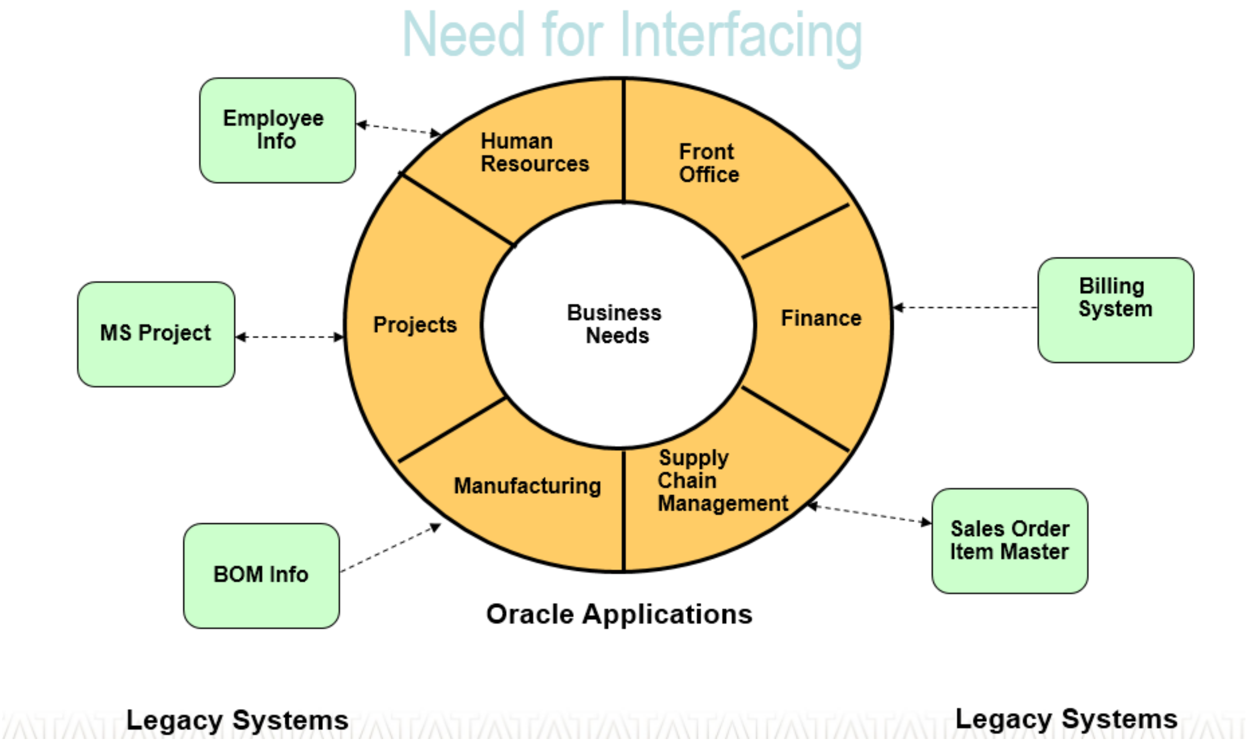
Note: We should keep in mind that Vendor/Supplier are entities to whom we place order for goods and services and pay them.

Customer/Buyer:

A party that receives or consumes products (goods or services).

Note: We should keep in mind that Customer/Buyer are the entities who place order with us for goods and services, and pay us.

Look at the diagram below, which shows a variety of legacy systems sharing data with Oracle Applications. They will be explained further.



Oracle Applications' various modules has to interact with the legacy system to send and receive data. In some cases, Oracle Applications comes with an in built interface, which can be used, and if it does not suit the need, a new Interfaces are developed to facilitate this. Needs of these interfaces are discussed below:

Human Resource <-> Employee Info:

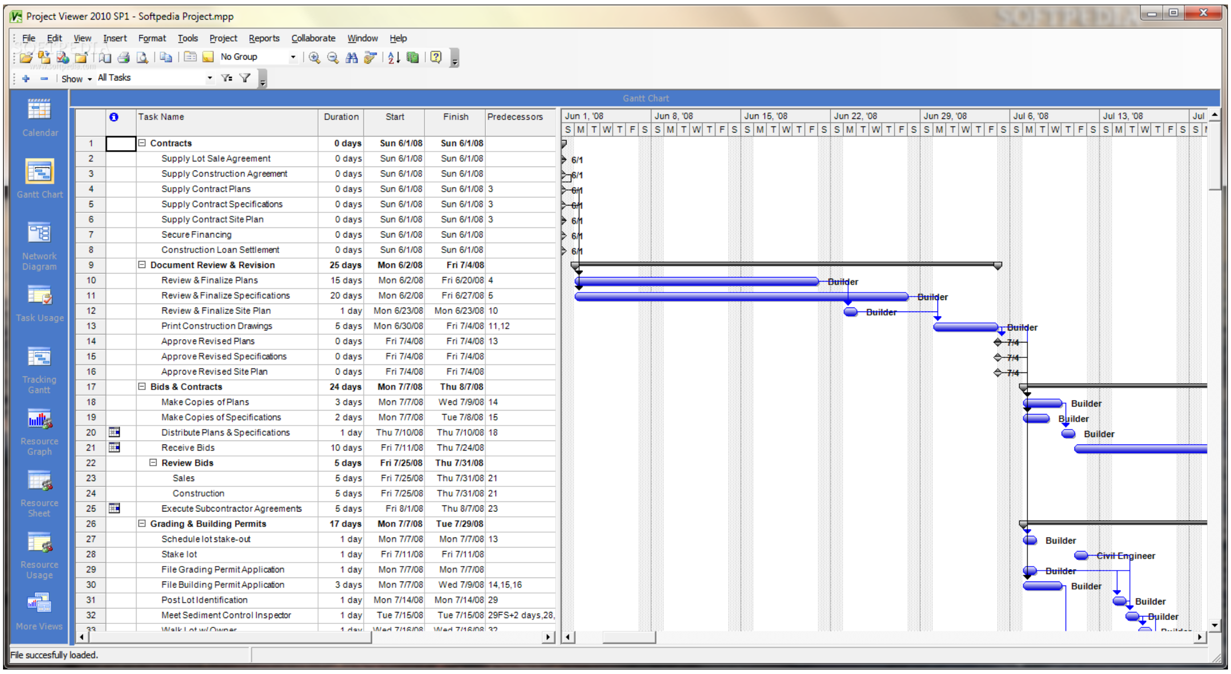
In many organizations, the Employee information is captured and stored in a separate legacy system (For example Zoho People, PeopleSoft or home grown system). Oracle Application Human Resource module needs to get the information from these systems. So, we develop an interface which may run at a regular interval, or may be driven by an event to push data into the Human Resource System. This interface will be called in **inbound interface.**

Again, when an employee it terminated, the termination will happen in the Human Resource system within Oracle Applications, but the data needs to be pushed back to the legacy Employee Info System, so we need to have an **Outbound interface.** This can also be a scheduled one or event driven, In this case the event will be Employee Separation.

Project <-> MS Project:

Projects module of Oracle Applications, helps project managers create and maintain project details inside oracle applications. Since the modules sits inside Oracle Applications, it has direct access to other modules(mainly Finance, Contracts in this case), a project maintained inside Oracle Applications has much more accurate information than a project managed with a third party tool.

Microsoft Projects is one such very popular tool for project management. This tool runs on your PC and is able to hold project details like the schedule, Cost, and is able to forecast information like Project End dates, Critical Path etc.



		Task Name	Duration	Start	Finish	Predecessors
1		⊟ **Contracts**	**0 days**	**Sun 6/1/08**	**Sun 6/1/08**	
2		Supply Lot Sale Agreement	0 days	Sun 6/1/08	Sun 6/1/08	
3		Supply Construction Agreement	0 days	Sun 6/1/08	Sun 6/1/08	
4		Supply Contract Plans	0 days	Sun 6/1/08	Sun 6/1/08	3
5		Supply Contract Specifications	0 days	Sun 6/1/08	Sun 6/1/08	3
6		Supply Contract Site Plan	0 days	Sun 6/1/08	Sun 6/1/08	3
7		Secure Financing	0 days	Sun 6/1/08	Sun 6/1/08	
8		Construction Loan Settlement	0 days	Sun 6/1/08	Sun 6/1/08	
9		⊟ **Document Review & Revision**	**25 days**	**Mon 6/2/08**	**Fri 7/4/08**	
10		Review & Finalize Plans	15 days	Mon 6/2/08	Fri 6/20/08	4
11		Review & Finalize Specifications	20 days	Mon 6/2/08	Fri 6/27/08	5
12		Review & Finalize Site Plan	1 day	Mon 6/23/08	Mon 6/23/08	10
13		Print Construction Drawings	5 days	Mon 6/30/08	Fri 7/4/08	11,12
14		Approve Revised Plans	0 days	Fri 7/4/08	Fri 7/4/08	13
15		Approve Revised Specifications	0 days	Fri 7/4/08	Fri 7/4/08	
16		Approve Revised Site Plan	0 days	Fri 7/4/08	Fri 7/4/08	
17		⊟ **Bids & Contracts**	**24 days**	**Mon 7/7/08**	**Thu 8/7/08**	
18		Make Copies of Plans	3 days	Mon 7/7/08	Wed 7/9/08	14
19		Make Copies of Specifications	2 days	Mon 7/7/08	Tue 7/8/08	15
20	▦	Distribute Plans & Specifications	1 day	Thu 7/10/08	Thu 7/10/08	18
21	▦	Receive Bids	10 days	Fri 7/11/08	Thu 7/24/08	
22		⊟ **Review Bids**	**5 days**	**Fri 7/25/08**	**Thu 7/31/08**	
23		Sales	5 days	Fri 7/25/08	Thu 7/31/08	21
24		Construction	5 days	Fri 7/25/08	Thu 7/31/08	21
25	▦	Execute Subcontractor Agreements	5 days	Fri 8/1/08	Thu 8/7/08	23
26		⊟ **Grading & Building Permits**	**17 days**	**Mon 7/7/08**	**Tue 7/29/08**	
27		Schedule lot stake-out	1 day	Mon 7/7/08	Mon 7/7/08	13
28		Stake lot	1 day	Fri 7/11/08	Fri 7/11/08	
29		File Grading Permit Application	1 day	Mon 7/7/08	Mon 7/7/08	
30		File Building Permit Application	3 days	Mon 7/7/08	Wed 7/9/08	14,15,16
31		Post Lot Identification	1 day	Mon 7/14/08	Mon 7/14/08	29
32		Meet Sediment Control Inspector	1 day	Tue 7/15/08	Tue 7/15/08	29FS+2 days,28,
33		Walk Lot w/ Owner	1 day	Wed 7/16/08	Wed 7/16/08	32

File succesfully loaded.

Project Managers find this tool very handy, and prefer to maintain the information in the tool. However the is tool needs to be synched with the project data inside Oracle Applications projects module, whenever a change is made in either places. For this purpose, in built interfaces come with Oracle Applications.

BOM Info <-> Manufacturing Module:

A Bills pf Material is a list of components which make a product. A BOM is the basic information needed for any manufacturing system. This data can be exported from the Manufacturing Module into a PL/SQL table, or a flat file, using the API - BOMPXINQ.EXPORT_BOM supplied by Oracle.

3. Interface – How does it work

Now that we have understood, what an interface can do, and why do we need them, let us see, how it works in real life.

There are several ways in which an interface can work, they can be categorized into following category:

1. Batch interface

   These interfaces are wither scheduled to run at a particular interval or are triggered by a program when a particular event occurs. These programs user variety of tools like Unix Shell script, Oracle Concurrent Manager, Unix Cron jobs, Oracle Advance Queue or scheduling tools like Control-M, Appworx.

2. Real time interface:

   Uses specialized tools like SOA / TIBCO , which publishes the information in real time and sync the source and target system. The system maintains a log of all the messages publishes and received, and makes sure that the target system has received the massage.

3. Datagram Service

   These interfaces are send messages over email. Since email uses datagram service, the data transfer depends on when the email message is received, which can depend on variety of conditions like – internet congestion, Mail Server load etc. Non critical data is transferred over this protocol.

4. Batch interface

Interfaces in this category uses one of the following tools:

a. Unix Shell scripting
b. SQL*Loader
c. PL/SQL programs

Shell Script:

A UNIX shell script is a command file created on Unix/Linux based operating system. A set of UNIX commands are put in a file called shell script. This shell script can be run at the command prompt. Shell scripting allows a comprehensive set of statements (if – then, while loops etc.) to write complex logic.

Let us see a simple example of shell script:

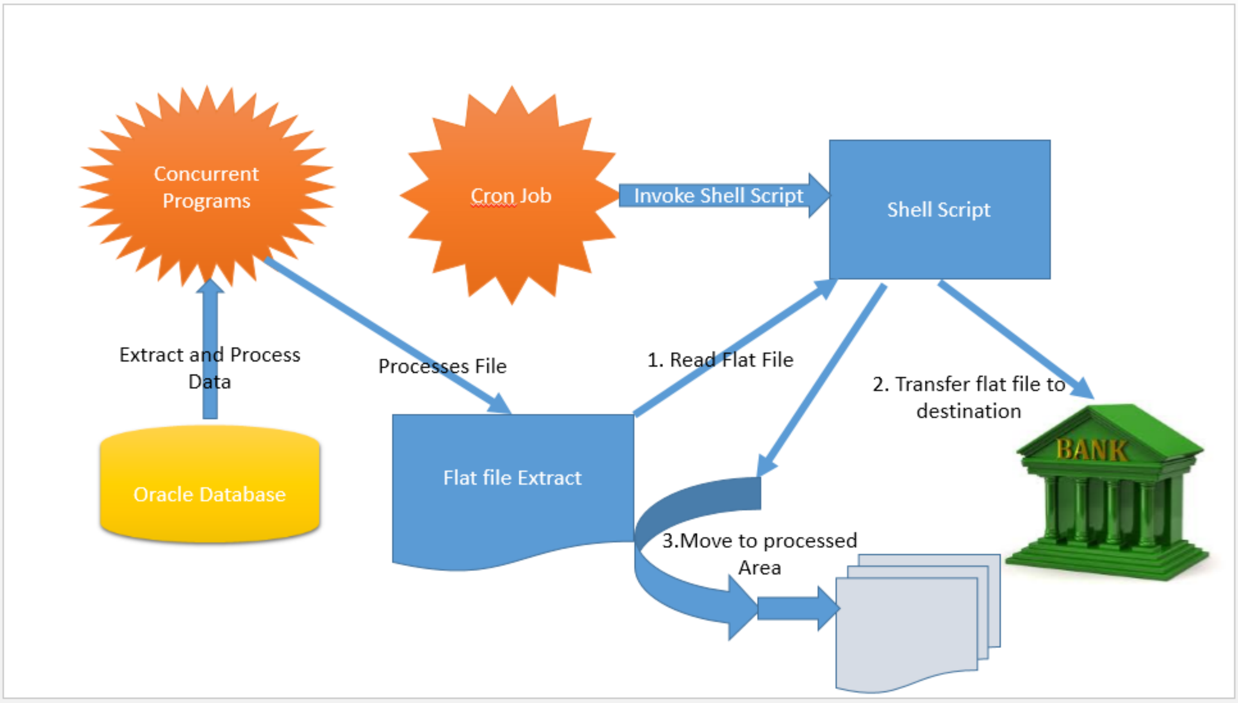
```
LOGON={yourlogonstring}
OUTPUT={youroutpufilename}
SCHEMA={yourschema name}
SRCTBL={yoursourcetablename}

sqlplus -s $LOGON <<EOF >$OUTPUT
 column {columnname1}      format {formatspec}  heading .
 column {columnname2}      format {formatspec}  heading .
 ...other columns...
 set linesize {line size needed}
 select {columnname1}
       ,{columnname2}
       ,...other columns...
 from $SCHEMA.$SRCTBL;
EOF
```

In the above example, the script accepts the oracle login id/password, the output file name, and the schema and table name and extracts the data inside this table to the flat file. This is the simplest example of an interface developed using UNIX shell scripting.

5. A Typical file based outbound interface

The below diagram shows a typical File based outbound interface at run:



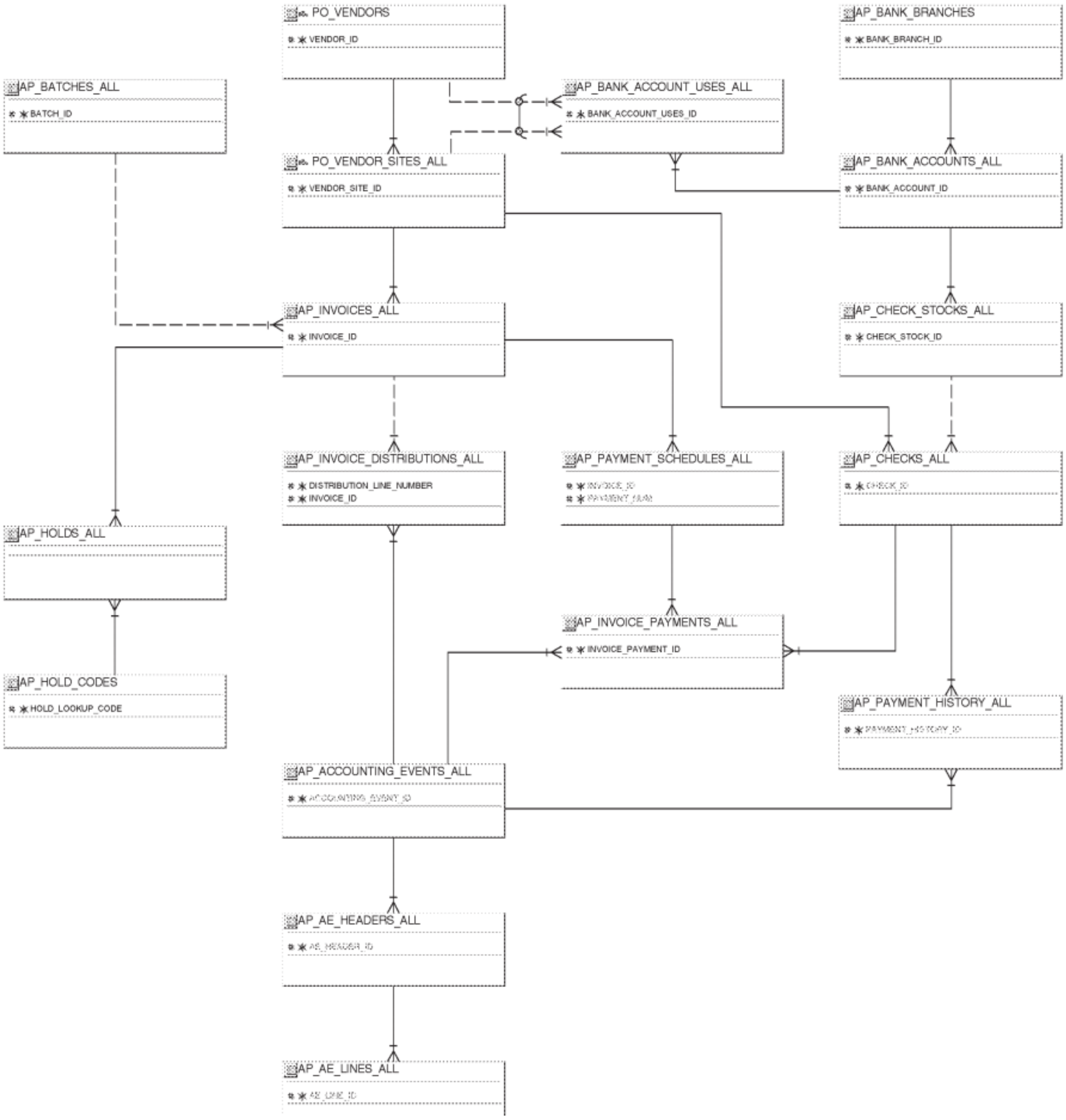
Let us understand how this works:

A concurrent program job runs within Oracle Applications pulling out information from the database. It messages the information by applying business logic, which can be quite complex in real situation. The concurrent program writes the processed information into a flat file on the operating system. There can be several Concurrent Programs running round the clock and producing files on the operating system.

The second piece is the cron jobs defined on the Operating system. A cron job is a job defined on the operating system m which can run automatically at the scheduled time. There are several cron jobs scheduled, normally one job is dedicated for one legacy system. This jobs keeps polling for files in a particular location of the Operating System. It transfers the file to the destination (in this case a bank), by using protocols like ftp or sftp , and then moves the processed file to another location designated for keeping processed archive files.

6. Oracle Open Interface Programs

The underlying table structure of oracle applications are quite complex. Tables are lined togather by complex referential integrity constriants. Below is the typical tables used for Invoice and payments, and their interrelationship.



So, if you need to load an invoice from a legacy system into Oracle Applications, you have to update different fields in all these tables, in very particular order, else your interface program will keep throwing millions of error.

Since it is almost impossible to update all the data in associated tables correctly, and thus also runs the chances of data corruption, Oracle never recommends tampering data into their core tables.

To ease this process, Oracle has given a set of **open Interface tables** for almost all its modules. These tables are quite simple with little validations. You are supposed to load the raw data into these tables, and the Oracle has supplied a set of **Open Interface Programs** for each module, which can pull out data from these interface tables and populate the same into appropriate oracle core tables. It also generates a list error records, which did not get processed during the import run. User can go back and correct the data and rerun the import program.

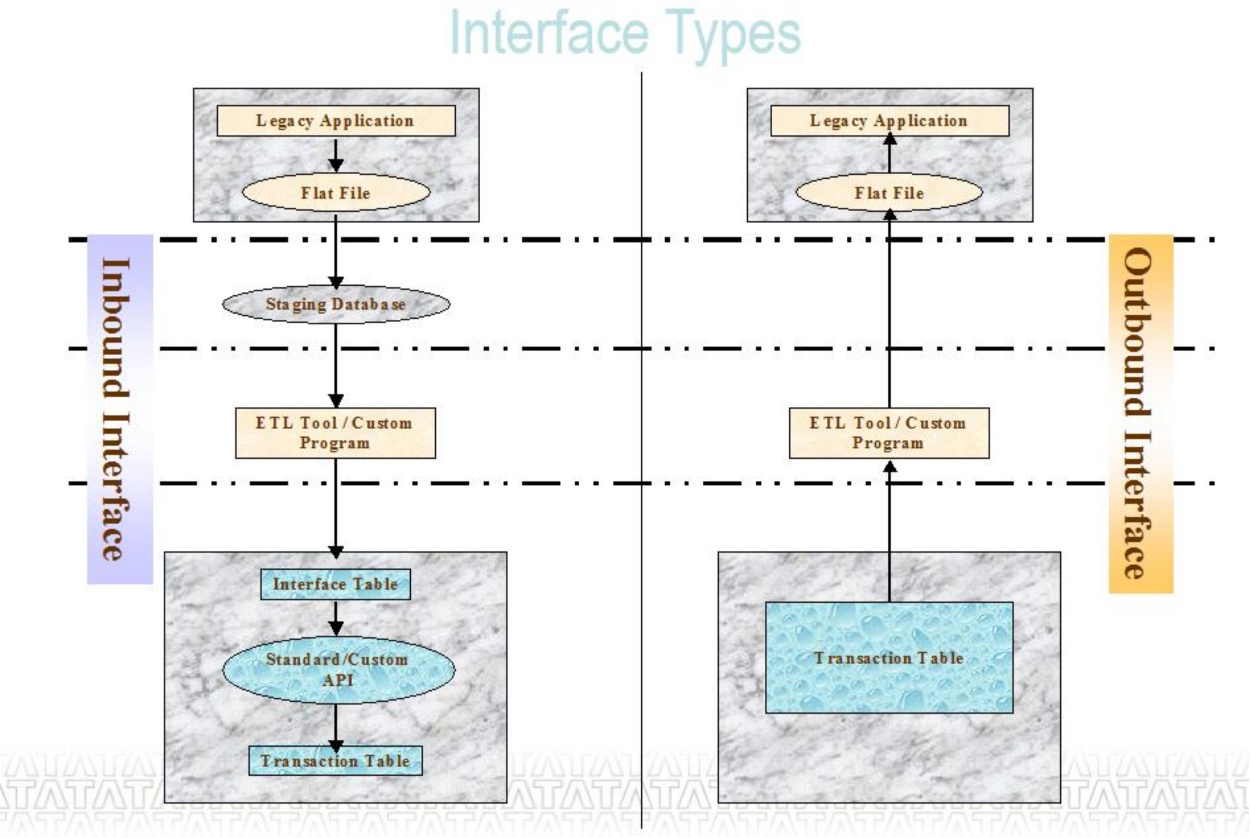
Al list of interface programs for each module is listed in Appendix A.

*Note: An Example of Open Interface Table for loading invoice data is AP_INVOICES_INTERFACE and AP_INVOICE_LINES_INTERFACE, which holds the Invoice header and Line details respectively, and the program '**Payables Open Interface Import'** imports the data to the transactional table.*

7. Use of staging Table

Using a staging table is a very popular way of storing intermediate data during processing.

We discussed in the previous section that Oracle provides open interface tables , which needs to be populated first and then Open interface programs can load data from these tables to core oracle tables. These tables are shown in the blue box in below diagram. Let us understand the events which happen during a typical **inbound interface** which loads data from a legacy system into Oracle Applications using the interface tables



.

Please refer to the left portion of the diagram for this discussion. Since Legacy systems may reside on variety of platform (Windows, VMS, Mac etc.), the simplest way to get the data is to dump the data at the source into a flat file. We use the tools available at the legacy platform to do this operation. Once the flat file is ready, the files is transported to the target operating system and then loaded into a **Staging Table.** The staging table has the format similar to the flat file, and is mostly created inside the oracle database of the Oracle Applications under a custom schema. This is done because:

1. It is much easier to message the data lying inside an Oracle table than a flat file.
2. Another reason for doing so is that the data needed to populate the Open interface tables may not be available in a single flat file, so having multiple staging tables makes it easy to combine the data and load into Open Interface tables.

Now we write custom interface programs to load data from the staging table(s) to the Open Interface tables. The custom interface program may encounter data errors while doing so. The error data dumped into an error table for further analysis. We often use TEL tools. ETL stands for **Extract, Transform and Load.** These tools are very useful in producing these custom interface codes, where we define the mapping between staging table columns and Open Interface table's columns and the tool generates the code for us. Some example of ETL tools are - Oracle Warehouse Builder, SAP Data Services, IBM Infosphere Information Server

After this, the API (Application Program interface) or Oracle Open Interface Programs are run to process the data and populate the transaction tables.

Let us now focus on the right side of the diagram – **Outbound Interface.** An outbound interface send the data from Oracle Applications to the Legacy System. Here we have to prepare the data in a format which suits the requirement of the legacy system, which can be heterogeneous. Here, we generate the flat file directly from the transactional tables (note that there is no need for the staging table here), either by writing the code, or generating using ETL tools. The file is sent to the legacy system, where it is uploaded into the legacy database using programs running at that end.

Note: Oracle AIM guidelines suggest to name custom schema like XXAP (starts with XX)...

Note: An API is a piece of code (function or procedure) provided by Oracle ,which perform a specific task.

TASK1: Write an inbound interface:

Scenario:

We have a supplier named ABC SUPPLIERS, who supplies A4 papers for our office use, the supplier is running a legacy system, and he wants to send the invoices in form of two text files. We need to write an interface program to load these invoices into our Payable System.

File1: Invoice_Header.txt:

Data:

```
101|'01-DEC-2014'|201|ABC SUPPLIERS|10|KOLKATA|29393.00|INR|OFFICE SUPPLIES FOR ABC
CORPORATION|1346752
```

Explanation of the data:

The invoice header details are dumped into this file, and the fields are separate by '|' symbol. The fields are arranged in the following sequence

```
INVOICE_ID|INVOICE_DATE|SUPPLIER_ID|SUPPLIER_NAME|SUPPLIER_SITE_ID|SUPPLIER_SIT
E_NAME|INVIOCE_TOTAL|INVOICE_CURRENCY|DESCRIPTION|PO_NUMBER|
```

This data will be uploaded into the table, whose table creation script is given below:

```
Create table STAGE_INV_HDR_<YourEmpNo>_AEN100 (
      INVOICE_ID             NUMBER(15)NOT NULL,
      INVOICE_DATE           DATE,
      SUPPLIER_ID            VARCHAR2(30),
      SUPPLIER_NAME          VARCHAR2(240),
      SUPPLIER_SITE_ID       NUMBER,
      SUPPLIER_SITE_NAME     VARCHAR2(30),
      INVIOCE_TOTAL          NUMBER,
      INVOICE_CURRENCY       CHAR(3),
      DESCRIPTION            VARCHAR2(240))
      PO_NUMBER              NUMBER)
```

File2: Invoice_line.txt

Data:

101|1|200|20|10|121.34|Bilt Premium Unruled A4 Multipurpose Paper

101|2|250|25|10|20344.89|Kodak High Gloss 210 mm x 297 mm Unruled A4 Photo Glossy Paper

Explanation of the data:

The invoice Line details are dumped into this file, and the fields are separate by '|' symbol. The fields are arranged in the following sequence:

INVOICE_ID|LINE_NUMBER|ITEM_QUANTITY|UNIT_PRICE|TAX_AMOUNT|NET PRICE

```
CREATE TABLE STAGE_INV_LINE_YourEmpNo_AEN100(
INVOICE_ID     NUMBER(15) NOT NULL,
LINE_NUMBER    NUMBER,
ITEM_QUANTITY  NUMBER,
UNIT_PRICE     NUMBER,
TAX_AMOUNT     NUMBER,
NET_PRICE      NUMBER,
LINE_ITEM_DESC VARCHAR2(240))
/
```

The Task:

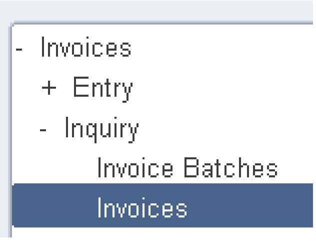
1. Copy the data of these two files and create the same file into the custom directory on Oracle Applications server (Check with your faculty for the location of the file, and assistance in creating the file).
2. Write a interface program (PL/SQL script), which will read these files, and populate a staging table.

3. Write another interface program (PL/Sql) to load the data from these staging tables into the Payables Open Interface tables - AP_INVOICES_INTERFACE and AP_INVOICE_LINES_INTERFACE. Once the data has been uploaded, update the filed PROCESSED_DATE with sysdate in both the staging tables.

4. Run the **Open Interface Programs** with responsibility – 'CB Payables , Vision Operations, USA'

5. Upon completion of the program, check the log to see, if the data was processed successfully.

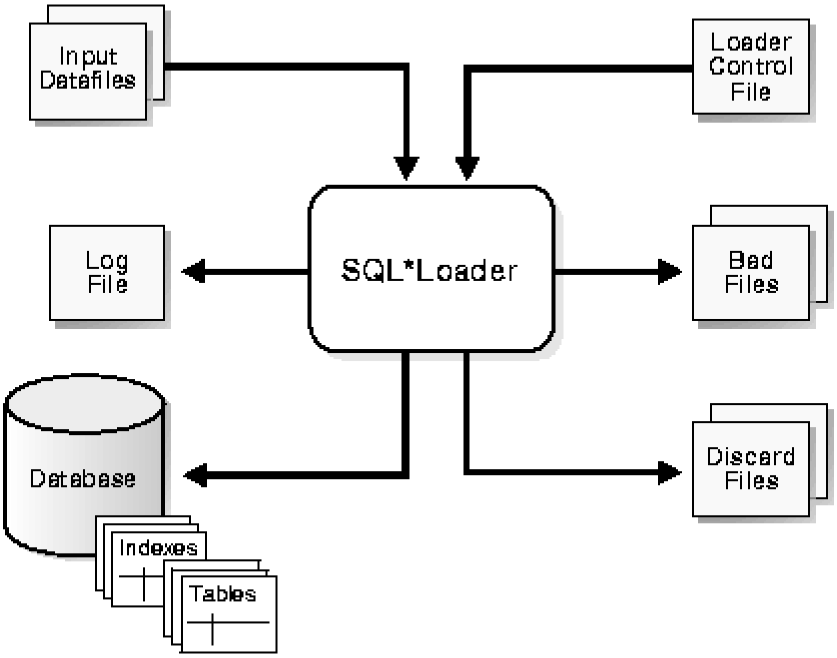
6. Go to the Invoice Query screen and query (Invoice > Inquiry > Invoices) and see, if the invoice has been crated in Oracle Applications.



7. Approve the invoice.

8. SQL*Loader

SQL*Loader is a simple yet powerful tool for uploading data from flat file into Oracle database tables. It uses simple command syntax.



The above diagram shows the function of sql*loader. It read from an input file, refers to the control file (for instructions), and loads data into Oracle tables. The rows which failed to be loaded are put into Bas and Discard files, and a log file is produced with summary of the operation, and errors encountered (if any).

Let us start by an example, here we will load data into an Oracle table from a flat file using simple sql*loader script.

Step1: Create the target table

```
SQL> create table employee

(

  id integer,
  name varchar2(10),
  dept varchar2(15),
  salary integer,
  hiredon date
)
```

Step2: Create the data

```
Vi employee.txt
100,Thomas,Sales,5000
200,Jason,Technology,5500
300,Mayla,Technology,7000
400,Nisha,Marketing,9500
500,Randy,Technology,6000
501,Ritu,Accounting,5400
```

Step3: Create the controlfile

```
$ vi example1.ctl
load data
 infile '/home/xx/employee.txt'
 into table employee
 fields terminated by ","
 ( id, name, dept, salary )
```

Step 4: Load data

```
$ sqlldr scott/tiger control=/home/xx/sqlldr-add-new.ctl
Commit point reached - logical record count 5
```

Verify the records are created in the database

Step 5: Verify loaded data

```
SQL> select * from employee;


        ID NAME       DEPT                SALARY HIREDON
---------- ---------- --------------- ---------- -------
       100 Thomas     Sales                 5000
       200 Jason      Technology            5500
       300 Mayla      Technology            7000
       400 Nisha      Marketing             9500
       500 Randy      Technology            6000
```

Step 5: Review the log

```
$ cat sqlldr-add-new.log
Control File:   /home/xx/sqlldr-add-new.ctl
Data File:      /home/xx/employee.txt

Table EMPLOYEE:
  5 Rows successfully loaded.
  0 Rows not loaded due to data errors.
  0 Rows not loaded because all WHEN clauses were failed.
  0 Rows not loaded because all fields were null.

Elapsed time was:     00:00:00.04
CPU time was:         00:00:00.00
```

We will limit our discussion on SQL*loader, for further detail of SQL*Loader, please refer oracle documentation.

9. Integration using SOA

So far we have seen that for every interface, we need to build some new programs, and any changes to the circumstances will need the programs to be rebuild. However in a dynamic environment, this becomes a cumbersome task. The traditional interface building process also suffers from lack of standardization, so any changes will need the developer/Architect to understand the current system, and in most case reverse engineer the code to get the design, and then rebuild the same. This brings a huge rework/testing and chances of introducing more bugs.

This calls for the need of **Enterprise Information Integration (EII)** and **Enterprise Application Integration (EAI)** products to create new interfaces in matter of days instead of months.

SOA brings a structured approach of building integration services. SOA stands for Service Oriented Architecture. SOA brings the following benefits:

1. Scalability
2. Availability
3. Greater software options
4. On-Demand Reporting
5. Security

To achieve this, SOA brings in a set of services/processes described below:

- **BPEL Process Manager** - Oracle BPEL Process Manager enables enterprises to integrate disparate applications and Web services into business processes using SOA (Service-oriented Architecture)

- **Oracle SOA Suite** - Oracle SOA Suite is a complete set of service infrastructure components for building, deploying, and managing SOAs. Oracle SOA Suite enables services to be created, managed, and orchestrated into composite applications and business processes.

- **Business Process Management** - Business process management is a management approach focused on aligning all aspects of an organization with the requirements of the units of that organization. It combines workflow and process technology with enterprise application integration functionality.

- **Oracle Data Integrator (ODI)** - Oracle Data Integrator is a comprehensive data integration platform that covers all data integration requirements: from high-volume, high-performance batch loads, to event-driven, trickle-feed integration processes, to SOA-enabled data services.

- **Oracle Enterprise Messaging Service: The** Oracle Enterprise Messaging Service provides a robust architecture for
integrating business-critical applications.

- **Oracle Enterprise Service Bus:** An ESB is an architecture that exploits Web services, messaging middleware, intelligent routing, and transformation. It must support request/response communication between loosely coupled SOA business components and one-way message delivery for sending notifications to event-driven business components. It must also allow more-complex message exchange patterns (MEPs).

- **Oracle Application Server B2B: Oracle** Application Server Integration B2B is an e-business integration product for the business-to-business (B2B) exchange of services, information, and products. If you know who you want to trade with (for example, a specific supplier), what you want to do (for example, send a purchase order), and how you want to do it (for example, send the purchase order over the Internet), then you have defined a basic B2B transaction.

- **Oracle Imaging and Process Management: Oracle** Imaging and Process Management is the most complete, integrated imaging solutions for end-to-end management of document images with coupling of company business processes. It leverages Oracle Document Capture and Oracle Distributed Document Capture for image capture, Oracle Forms Recognition for intelligent data capture, and provides annotation and markup of images, automates routing and approvals, and a scalable repository supporting enterprise-wide applications. With this companies can quickly automate business processes in Oracle E-Business Suite, PeopleSoft Enterprise, and JD Edwards EnterpriseOne.

- **Oracle Service Registry** - is a UDDI v3 (Universal Description Discovery and Integration version number 3) registry with an embedded governance framework. It provides a repository where services can be registered and reused for developing or modifying applications.

- **Oracle Web Services Manager (OWSM):** Oracle Web Services Manager offers a comprehensive and easy-to-use solution for policy management and security of service infrastructure. With Oracle WSM companies allows to

    1. centrally define and store declarative policies applied to the multiple web services making up a SOA infrastructure,

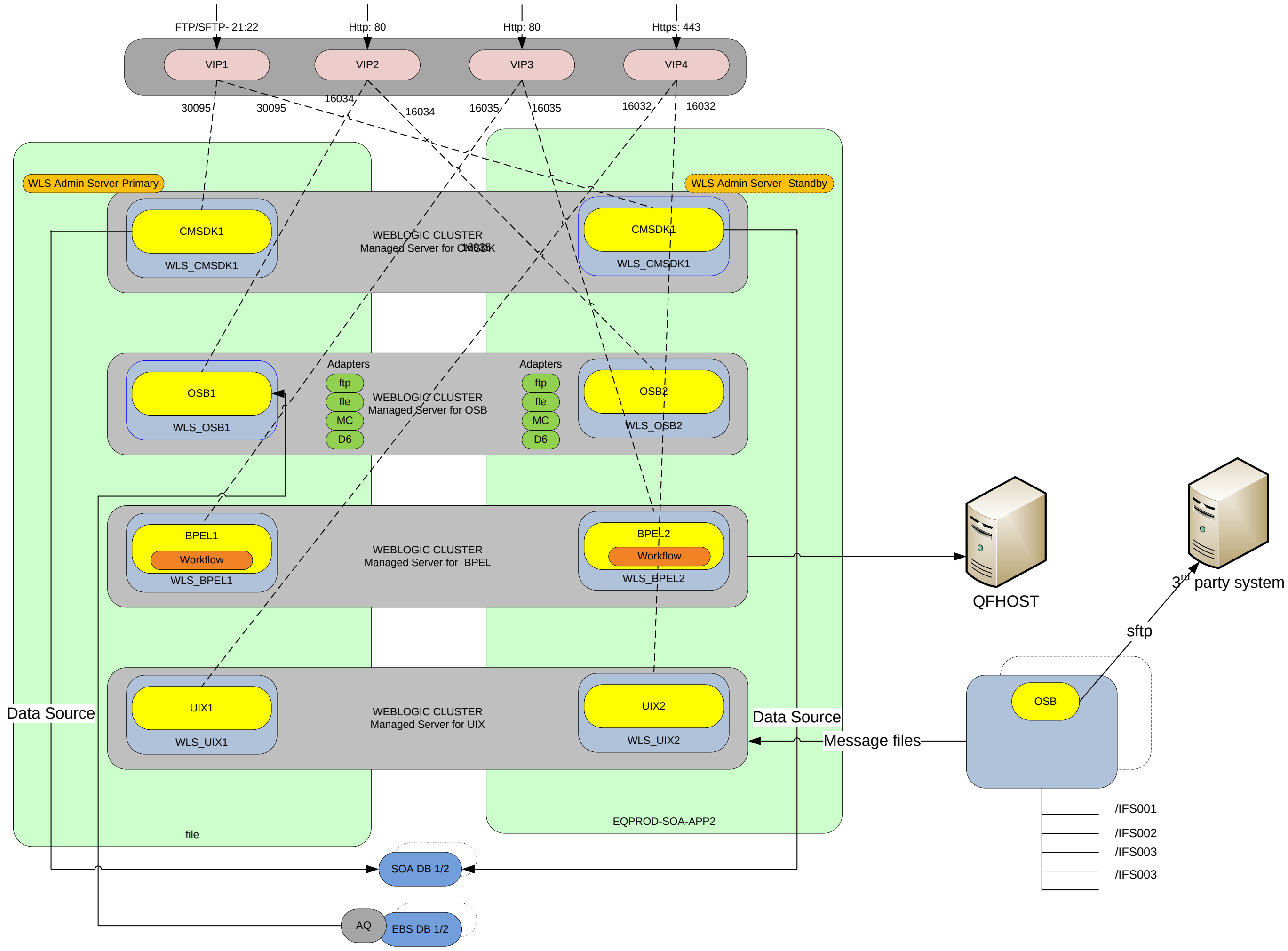
    2. locally enforce security and management policies through configurable agents, and

    3. Monitor runtime security events such as failed authentication or authorization.

10. Integration using SOA – A Case study

This section illustrates a real life example of SOA based integration for Oracle Applications suite integrated with around 200 third party systems. Detailed discussion of SOA implementation is outside the scope of this course.

SOA application tier services will be deployed on a pair of redundant servers (soa-app1 and soa-app2), and the database will reside on RAC instance built on soa-db1 and soa-db2. WebLogic cluster will be control the Managed servers. SOA application tier will have following components:



1. Weblogic server cluster: Four Managed server of Weblogic server (CMSDK, OSB, BPEL and UIX) will be created on each of application tiers. Eqprod-soa-app1 will have Weblogic admin server deployed, which will control the members in the managed server. The SOA domain will have one Admin server and two managed servers.
2. In the event of failure of Weblogic Admin server on the primary app tier, pre-configured admin server will be started manually, Load balancer will be running a probe on each server to sense component failure, and will redirect the new connections to the services on the active node. However the connections already established at the time of failure will need to be reinstated.
3. CMSDK services will run on higher port 30095 on each ap tier. CMSDK folders will be created in the IFSYS schema of the SOA database , which will run on RAC on node – eqprod-db1 and eaprod-db2
4. OSB services will be running on each host on higher port 16034 under its managed server. This will have different kind of file adapters.
5. BPEL will run on the BPEL managed server. This will host all the workflows.
6. UIX services will run on the UIX managed server, and will perform the function of transferring the flat files to the target servers (non EBS internal applications and 3rd party servers)

The following diagram describes the SOA components described above:

Data flow:

The inbound and outbound information flow from/to SOA will happen in the following way:

Outbound:

EBS will put the outgoing data into BOLINF queue. OSB will keep polling this queue and appropriate BPEL workflow will be initiated. Data will be encrypted as per the UIX configuration defined for the interface. If the interface is directed for QFHOST, an ftp connection will established from BPEL. If is it is directed for third party, it will be directed to the sftp server, which will direct the data to the appropriate third party system

Inbound:

Data coming from QFHOST:

QFHOST will connect to CMDSK VIP1 and poll the CMSDK folder for the particular interface. CMSDK will put the data in the SOA database queue. OSB will poll the data from the queue and pass it on to the appropriate BPEL workflow. The workflow will decrypt the data (if applicable), and pass it on to the EBS BOLINF queue.

Data coming from Third Party:

Data coming from third party systems fall under two categories:

SOA Polling Third Party Systems

SOA OSB adapters running on sftp server connects to the third party systems and polls the data and puts into the CMSDK folder. CMSDK will put the data in the SOA database queue. OSB will poll the data from the queue and pass it on to the appropriate BPEL workflow. The workflow will decrypt the data (if applicable), and pass it on to the EBS BOLINF queue.

Third Parties Connecting to SOA and pushing the files:

Third party will connect to SOA OSB running on the sftp server and push the files to the sftp server, which will be forwarded to the CMSDK folder. CMSDK will put the data in the SOA database queue. OSB will poll the data from the queue and pass it on to the appropriate BPEL workflow. The workflow will decrypt the data (if applicable), and pass it on to the EBS BOLINF queue.

TASK2: Write an outbound interface:

1. Introduction

Business Event System

Business Event System (BES) provides the mechanism for customers to build integration with Other applications and systems such as Oracle BPEL Process Manager in an easy, non-invasive Manner. BES consists of the Event Manager, which lets you register subscriptions to significant Events, and event activities, which let you model business events within workflow processes. Subscriptions can include custom logic in both Java and PL/SQL.

XML Gateway
Oracle XML Gateway is a set of services that allows easy integration with the Oracle E-Business Suite to support XML messaging.
Oracle XML Gateway consumes events raised by the Oracle E-Business Suite and subscribes to inbound events for processing. Oracle XML Gateway uses the message propagation feature of Oracle Advanced Queuing to integrate with the Oracle Transport Agent to deliver messages to and receive messages from business partners. Oracle XML Gateway supports both Business-to-Business (B2B) and Application-to-Application (A2A) initiatives. B2B initiatives include communicating business documents and participating in industry exchanges. An example of an A2A initiative is data integration with legacy and disparate systems.

With Oracle XML Gateway services, you are assured consistent XML message implementation when integrating with the Oracle E-Business Suite, thereby lowering integration costs and expediting message implementation while supporting corporate e-business initiatives.

Oracle e-Commerce / EDI Gateway

Oracle Applications provides users with the ability to conduct business electronically between trading partners based on Electronic Commerce standards and methodology. One form of Electronic Commerce is Electronic Data Interchange (EDI). EDI is an electronic exchange of data between trading partners. Interface data files are exchanged in a standard format to minimize manual effort, speed data processing, and ensure accuracy. The Oracle e-Commerce Gateway (formerly known as Oracle EDI Gateway) performs the following functions:

- Define trading partner groups and trading partner locations.

- Enable transactions for trading partners.

- Provide general code conversion between trading partner codes or standard codes and the codes defined in Oracle Applications.

- Define interface data files so that application data can integrate with your trading partner’s application or an EDI translator.

- For inbound transactions, import data into application open interface tables so that application program interfaces (API) can validate and update Oracle application tables.

- For outbound transactions, extract, format, and write application data to interface data files.

Oracle e-Commerce Gateway augments the existing standard paper document capabilities of Oracle Applications, or adds functionality where no corresponding paper documents exist.

Concurrent Program / Interface Tables

In Oracle Applications, concurrent processing simultaneously executes programs running in the background with online operations to fully utilize your hardware capacity. You can write a program (called a”concurrent program”) that runs as a concurrent process. Typically, you create concurrent programs for long-running, data- intensive tasks, such as posting a journal or generating a report.

PeopleSoft Integration Tools

Integration Broker

PeopleSoft Integration Broker is a middleware technology that facilitates synchronous and asynchronous messaging among internal systems and trading partners, while managing message structure, message format, and transport disparities.

PeopleSoft Integration Broker comprises two high-level subsystems: the Integration Engine and the Integration Gateway. The Integration Engine runs on the PeopleSoft application server. It is tied closely to PeopleSoft applications and produces or consumes messages for these applications.

The Integration Gateway is a platform that manages the actual receipt and delivery of messages passed among systems through the PeopleSoft Integration Broker. It provides support for the leading TCP/IP protocols used in the marketplace today, and more importantly, provides extensible interfaces for the development of new connectors for communication with legacy, ERP, and internet-based systems.

PeopleTools

PeopleTools refers to the collection of proprietary PeopleSoft tools for the development and execution of applications.

Component / Service Interface – APIs

A component interface is a set of application programming interfaces (APIs) that you can use to access and modify PeopleSoft database information using a program instead of the PeopleSoft client. PeopleSoft Component Interfaces expose a PeopleSoft component (a set of pages grouped for a business purpose) for synchronous access from another application (PeopleCode, Java, C/C++, or Component Object Model [COM]). A PeopleCode program or an external program (Java, C/C++, or COM) can view, enter, manipulate, and access PeopleSoft component data, business logic, and functionality without being online.

Application Message

Application Messages, or Message definitions, are the heart of PeopleSoft Integration Broker. They are templates for the data that the application sends or receives. Message definitions are created in PeopleSoft Application Designer. There are two types of PeopleSoft messages:

• Rowset-based (Structured) messages. For hierarchical data based on PeopleSoft records, these messages are created by assembling records, organizing them into a hierarchy, and selecting fields from those records to include in the message. The result is a rowset that doesn’t need to match an existing rowset structure in the application. PeopleCode Rowset and Message classes are used to generate, send, receive, and process these messages.

• Non-rowset-based (Unstructured) messages. These messages can have virtually any structure and content. You create a message definition, but you do not insert any records. The message definition serves as a placeholder for the actual message. PeopleCode XmlDoc and Message classes are used to generate, send, receive and process these messages. You can also use the PeopleCode SoapDoc class to generate and process these Messages for web services and SOAP messages.

File Layout

A File Layout object is a definitional map between a file (fixed format, comma separated values or XML) and a PeopleSoft Record definition. Creating File Layout objects requires just a few mouse clicks once you have a Record definition and sample file. Once you have created a File Layout object in PeopleTools 8.42 and higher, you can click the AE button in the Application Designer to have PeopleTools write the Application Engine PeopleCode for you that will load the file from the file server into PeopleSoft Records. For more information and examples see the Enterprise PeopleTools 8.46 PeopleBook: PeopleSoft Application Designer > Understanding File Layouts.

JD Edwards Web Services Gateway Components

Broker

The broker is a high-speed message router. It is the primary component of the platform's messaging facility. It uses a publish-and-subscribe model for asynchronous processing.

Integration Server

The integration server is the platform's central runtime component and the primary engine for the execution of the integration logic.

Integration Point

An Integration Point acts as the connection between the physical implementation and the logical business function. One can say that a system exposes its business functionality as Integration Points.

White Paper Title Page 10

XBP

An XBP is a PeopleSoft marketing term, denoting specific Integration Points in the PeopleSoft EnterpriseOne product family applications that have been dubbed as such. Not all Integration Points exposed by PeopleSoft EnterpriseOne applications are XBP's.

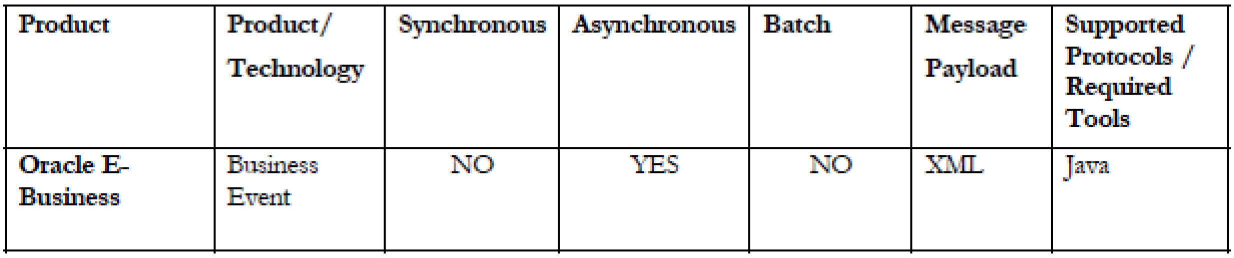
Canonical

Canonical events are generic representations of data (both data structure and data content) created and published by a source integration within XPI Enterprise Integrator and subscribed to by one or more target integrations.

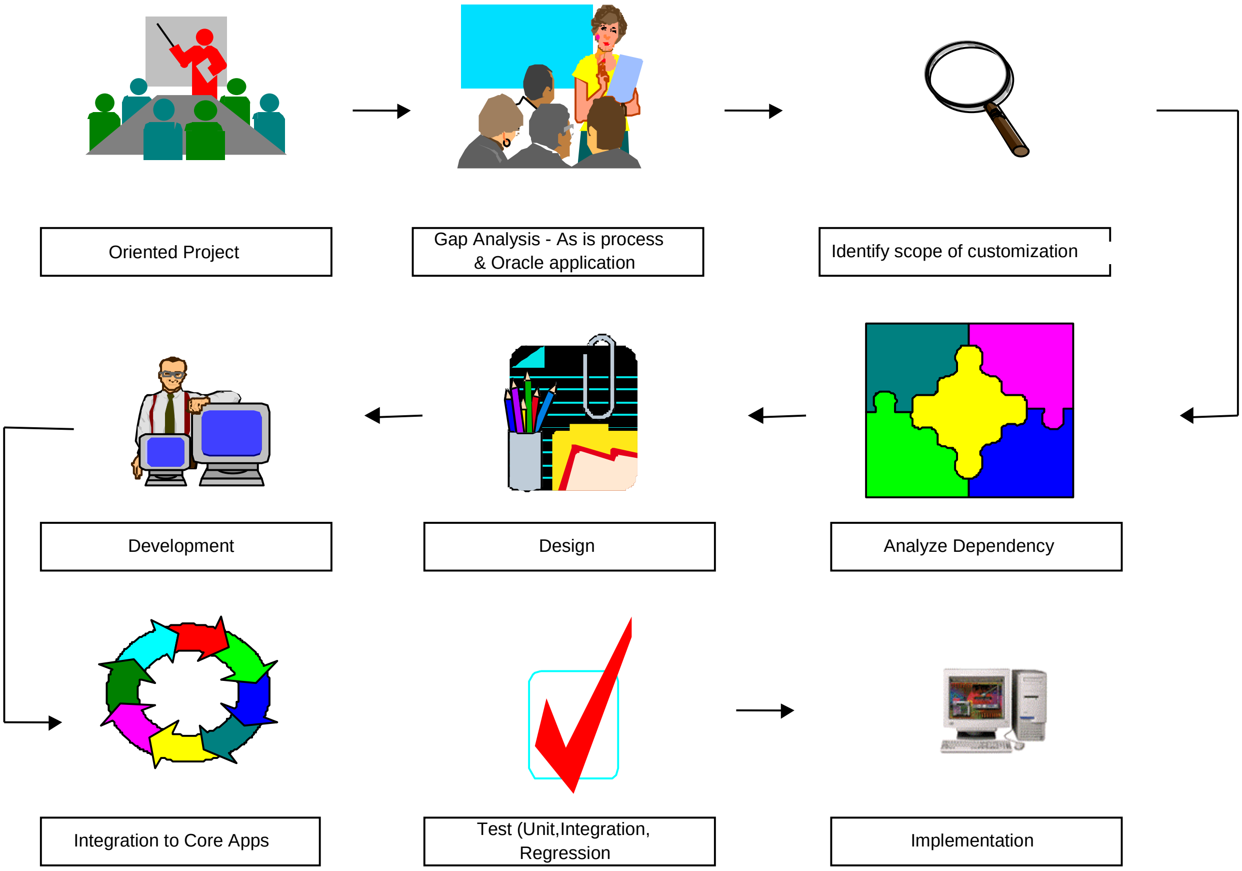
Cross-Application Integration Using BPEL

The table below provides a mapping of technology choices available in each product for a given integration pattern that can be orchestrated using BPEL PM.

Table:



Product	Product/ Technology	Synchronous	Asynchronous	Batch	Message Payload	Supported Protocols / Required Tools
Oracle E-Business	Business Event	NO	YES	NO	XML	Java



Oriented Project

Gap Analysis - As is process & Oracle application

Identify scope of customization

Analyze Dependency

Design

Development

Integration to Core Apps

Test (Unit,Integration, Regression

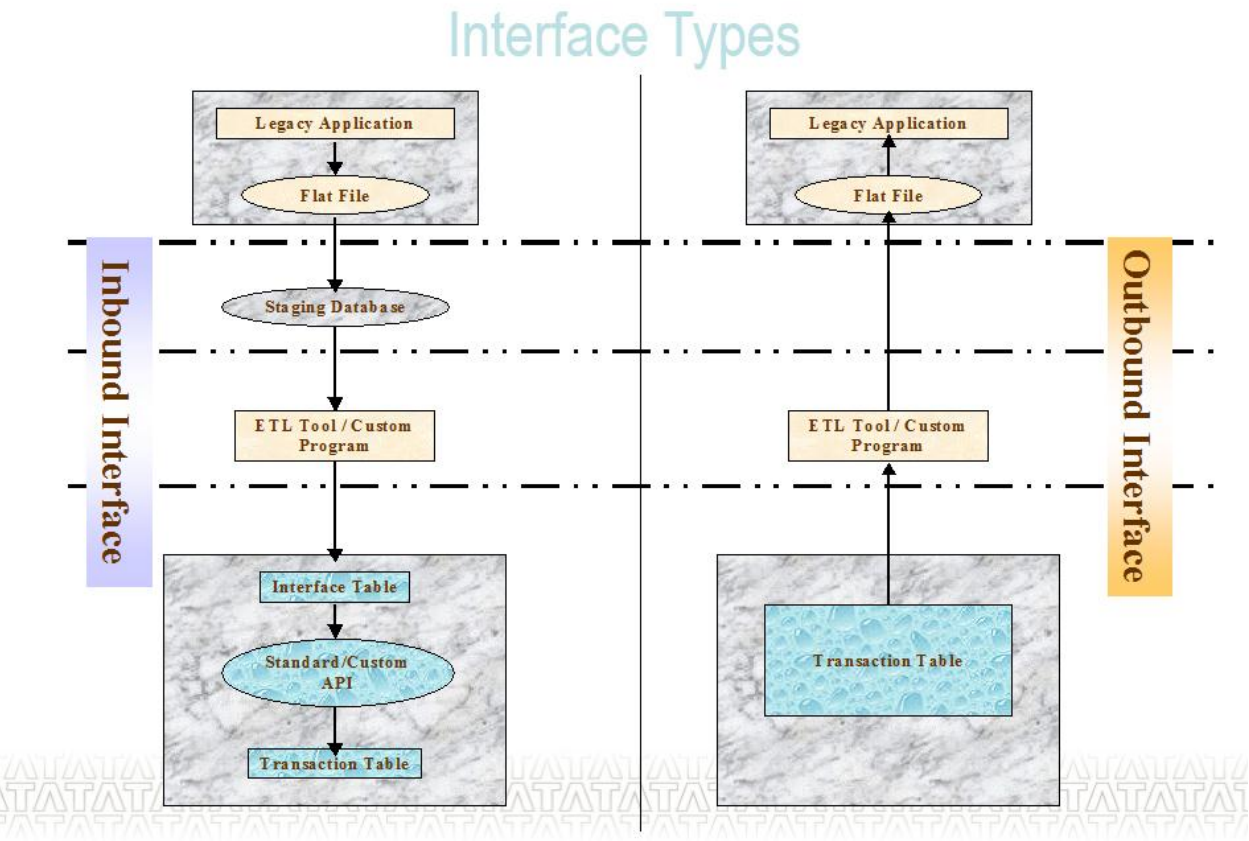
Implementation

Objective

The Purpose of the Interface & Conversion is to serve as a bridge between the Oracle applications and the legacy system.

The Core Process Analysis constitutes four broad level activities

- Data Migration/Conversion: Replacement of legacy application with the Oracle Apps.
- Uni/Bi-directional data flow between legacy and Oracle Application.



Appendix A: Oracle APIs and Open Interfaces

Oracle Purchasing:

1. Requisitions Open Interface
2. Purchasing Documents Open Interface
3. Cancel PO APIs
4. Receiving Open Interface

Oracle Inventory:

1. Open Transaction Interface
2.1 Customer Item Interface
2.2 Open Item Interface
2.3 Cycle Count Open Interface
3.1 Open Replenishment Interface
3.2 Reservations Open Interface
3.3 Move Orders Open Interface

OM:

1. Order Import
2. Process Order API
3. RLM Open Interfaces

Actions, APIs, and Parameters: Descriptions of the APIs used for various functions and the API parameters.
Application Parameter Initialization: Description of the application parameter initialization call.

Trip API: Create and update trip records and perform actions on trips.

Stop API: Create and update stop records and perform actions on stops.

Deliveries API: Create and update trip stop records and perform actions on trip stops.

Delivery Details API: Assign and unassign delivery details to and from deliveries, split a delivery detail, update a delivery detail with new information, and create trips and deliveries for multiple delivery lines.

Container API: Create container records, update container records, autopack containers, perform actions on containers.

Freight Cost APIs: Create freight cost records, update freight cost records, validate freight cost types, and delete freight cost records.

Tables

OE_ORDER_HEADERS_ALL
OE_ORDER_LINES_ALL
WSH_DELIVERY_DETAILS
OE_ORDER_HOLDS_ALL
OE_PRICE_ADJUSTMENTS
OE_TRANSACTION_TYPES_ALL
OE_DROP_SHIP_SOURCES
OE_SETS
OE_SYSTEM_PARAMETSR
MTL_DEMANDS
MTL_RESRVATIONS

Inventory Open Transaction Interface:

Oracle Inventory provides an open interface for you to load transactions from external applications and feeder systems. These transactions could include sales order shipment transactions from an Order Management system other than Oracle Order Management, or they could be simple material issues, receipts, or transfers loaded from data collection devices. The following transaction types are supported by this interface:

- Inventory issues and receipts (including user-defined transaction types)
- Subinventory transfers
- Direct interorganization transfers

- Intransit shipments
- WIP component issues and returns
- WIP assembly completions and returns
- Sales order shipments
- Inventory average cost updates
- LPN Pack
- Unpack
- Split Transactions
- Inventory Lot Split/ Merge/ Translate Transactions

This interface is also used as an integration point with Oracle Order Management for shipment transactions. Oracle Order Management's Inventory Interface program populates the interface tables with transactions submitted through the Confirm Shipments window.

You must write the load program that inserts a single row for each transaction into the MTL_TRANSACTIONS_INTERFACE table. For material movement of items that are under lot or serial control, you must also insert rows into MTL_TRANSACTION_LOTS_INTERFACE and MTL_SERIAL_NUMBERS_INTERFACE respectively. If you insert WIP assembly/completion transactions that complete or return job assemblies, you must also insert rows into the CST_COMP_SNAP_ INTERFACE table if the organization referenced uses average costing. The system uses this information to calculate completion cost.

There are two modes you can use to process your transactions through the interface. In the first processing mode, you populate the interface table only. Then the Transaction Manager polls the interface table asynchronously looking for transactions to process, groups the transaction rows, and launches a Transaction Worker to process each group.

In the second processing mode, you insert the rows in the interface table and call a Transaction Worker directly, passing the group identifier of the interfaced transactions as a parameter so that the worker can recognize which subset of transactions to process.

The Transaction Worker calls the Transaction Validator, which validates the row, updates the error code and explanation if a validation or processing error occurs, and derives or defaults any additional columns. Next, the Transaction Processor records the transaction details in the transaction history table along with relevant current cost information. All material movement transactions update inventory perpetual balances for the issue, receipt, or transfer locations. Once the transaction has been successfully processed, the corresponding row is deleted from the interface table. Finally, the transaction is costed by the transaction cost processor, which runs periodically, picking up all transactions from the history table that have not yet been marked as costed.

Open Replenishment Interface

Oracle Inventory provides an open interface for you to easily load replenishment requests from external systems such as a barcode application. Such requests may be in the form of stock-take counts or requisition requests for subinventories in which you do not track quantities.

Cycle Count Entries Interface

You can import cycle count entries from an external system into Oracle Inventory using the Cycle Count Entries Interface. This interface validates all data that you import into Oracle Inventory. It also performs foreign key validation and checks for attribute inter-dependencies, acceptable values, and value ranges. The interface ensures that the imported cycle count entries contain the same detail as items entered manually using the Cycle Count Entries window. Errors detected during validation are written to the Cycle Count Interface Errors table.

Kanban Application Program Interface

The Kanban API is a public API that allows you to update the supply status of kanban cards. To accomplish this task, you use the public procedure update_card_supply_status

Item Open Interface

You can import items from any source into Oracle Inventory and Oracle Engineering using the Item Open Interface. With this interface, you can convert inventory items from another inventory system, migrate assembly and component items from a legacy manufacturing system, convert purchased items from a custom purchasing system, and import new items from a product data management package. The Item Open Interface validates your data, ensuring that your imported items contain the same item detail as items that you enter manually in the Master Item window.

You can also import item category assignments. This can be done simultaneously with a process of importing items, or as a separate task of importing item category assignments only. For this purpose, the Inventory menu contains the Import submenu with the Import Items and Import Item Category Assignments menu entries.

Receiving Open Interface

You use the Receiving Open Interface to process and validate receipt data that comes from sources other than the Receipts window in Oracle Purchasing. These sources include:

- Receipt information from other Oracle applications or legacy systems
- Brocades and other receiving information from scanners and radio frequency devices
- Advance Shipment Notices (ASNs) from suppliers

The Receiving Open Interface maintains the integrity of the new data as well as the receipt data that resides in Oracle Purchasing. The Receiving Open Interface does not support:

- Movement statistics
- Dynamic locators

BOM

Bills of Materials Open Interfaces

WIP

Open Move Transaction Interface

You can load Move transaction information into the Open Move Transaction Interface from a variety of sources, including external data collection devices such as bar code readers, automated test equipment, cell controllers, and other manufacturing execution systems. You then use the interface to load these transactions into Oracle Work in Process. All transactions are validated and invalid transactions are marked, so that you can correct and resubmit them.

Open Resource Transaction Interface

You can use external data collection devices such as bar code readers, payroll systems, and time card entry forms to collect resource and overhead transaction data, then load the data into the Open Resource Transaction Interface for Oracle Work in Process to process.

Work Order Interface

The Work Order Interface enables you to import Discrete job and Repetitive schedule header information, and Discrete job operations, material, resource, and scheduling information from any source, using a single process.

You can import:

- Planned orders for new Discrete jobs,
- Discrete job operations, components, resources, resource usage, and scheduling details
- Update and reschedule recommendations for existing Discrete jobs
- Suggested Repetitive schedules Work in Process then uses this information to automatically create new Discrete jobs and pending Repetitive Schedules, or to update existing Discrete jobs.

MRP

Open Forecast Interface

You can import forecasts from any source using the Open Forecast Interface table. Oracle Master Scheduling/MRP automatically validates and implements imported forecasts as new forecasts in Oracle Master Scheduling/MRP.

Cost Management

Periodic Cost Open Interface

The Oracle Periodic Cost Open Interface provides an open interface for you to easily load periodic item costs from external applications or legacy systems and migrate them into the Oracle Cost Management Application. This interface should only be used to bring in periodic costs for the first opened periodic period. It cannot be used for subsequent periods. Costs in subsequent periods are calculated by the system.

Cost Import Interface

The Oracle Cost Import Interface enables you to import costs for items from legacy systems, and import new cost information for existing items. Importing resource costs and resource overhead rates is also supported. You will also be able to replace existing cost information with the new cost information. However, updating of existing costs is not supported. Importing costs into frozen cost type is also not supported. The item costs will have to be imported into another cost type and then the cost update may be run to update the frozen cost type

OM

Order Import
Pricing Open interface
Pick release