IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.102 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 14, Issue 3, March 2025

DOI: 10.17148/IJARCCE.2025.14309

Utilizing SOA for Product-Item Master (Oracle EBusiness Suite – Seibel CRM) Integration

Sadia Tahseen

Fusion Middleware Architect/Developer, USA

Abstract: This paper is a case study of a project done at a manufacturing company wherein the project goal was to do Integration between Oracle Ebusiness Suite and Seibel CRM. This paper talks about the approach taken in this integration

Term	Definition
EIM	Siebel Enterprise Integration Manager. It is a server component in the Siebel EAI component group that transfers data between the Siebel database and other data sources in the company.
EAI	Siebel Enterprise Application Integration. Siebel EAI includes a set of tools, technologies, and prebuilt functional integrations that facilitate application integration with Siebel and other systems.
SQL*Loader	This is an Oracle-supplied utility that allows data to be loaded from flat files into one or more database tables.
Siebel Workflow	A Siebel Workflow is a series of Siebel Business Services, processes or operations linked together to accomplish a business task.
Import Object	Is located in Siebel tools. Configures the parameters of a Siebel Business Component Object entity which supports client side import.
Conversion Object	Conversion Object also commonly referred to as “IO”, is an object used to define or explain Siebel Objects such as Business Component and Business Objects. An “IO” is an object providing meta data about Siebel Objects in form of XML. The primary use of a Conversion Object is for Conversion with external applications but is not limited to that.
CSV or (.csv)	Comma Separated Value. A file format that separates its elements with a comma. Data engine that are equipped to handles a .csv format can parse its contents into columns
Web Service	Is a Siebel Application component that communicates via open protocols (HTTP, SMTP, ETC) and processes XML message frames using SOAP. The Web Service describes its messages using XML schema and itself using a WSDL.
EIM Table	Siebel EIM tables are intermediate database tables that act as a staging area between the base tables in the Siebel database and other databases. EIM tables are designed to be simple and straightforward so they can be loaded or read by way of external programs.
Integration Object	Integration Object also commonly referred to as “IO”, is an object used to define or explain Siebel Objects such as Business Component and Business Objects. An “IO” is an object providing meta data about Siebel Objects in form of XML. The primary use of an Integration Object is for integration with external applications but is not limited to that.
Integration Component	A constituent part of a Siebel integration object. A metadata structure that represent business component and their fields. Each integration component can also have child integration components and fields.
Business Service	A business service is a reusable module containing a set of methods. It provides the ability to call its C++ or script methods from customer-defined scripts and object interface logic, through the invoke-method mechanism.
IFB	An EIM configuration file that resides in the Siebel Server/admin directory. EIM reads this configuration file which specifies the EIM process to perform (import, update, merge, delete, or export) using the appropriate parameters.
SOA	Service Oriented Architecture. Oracle SOA Suite simplifies connectivity by providing a unified experience to integrate across applications.
BPEL	Oracle BPEL Process Manager enables you to orchestrate synchronous and asynchronous services into end-to-end BPEL process flows.
Mediator	Oracle Mediator is used to route, validate, filter and transform and create rules for them.
Adapters	Oracle Adapters use JCA technology to connect external systems to the Oracle SOA Suite. Oracle SOA Suite provides the following technology adapters BAM, FTP, Java Messaging Service (JMS),Advanced Queuing (AQ),Files, Message Queuing (MQ) Series

PROBLEM STATEMENT

This design solves for order to cash Integration for Product Interface for manufacturing /retail/consumer businesses leveraging Oracle SOA 11g as middleware from Oracle EBS 11i/R12 to Siebel CRM 8 systems.The other devices/systems in field of invention don’t work well as proper middleware was not utilized and proper deployment methodology was not utilized .Some of failed methods are O2C PIPs and just using Foundation packs of AIA framework. Oracle SOA is preferable middleware layer for easy flow of data across these systems.

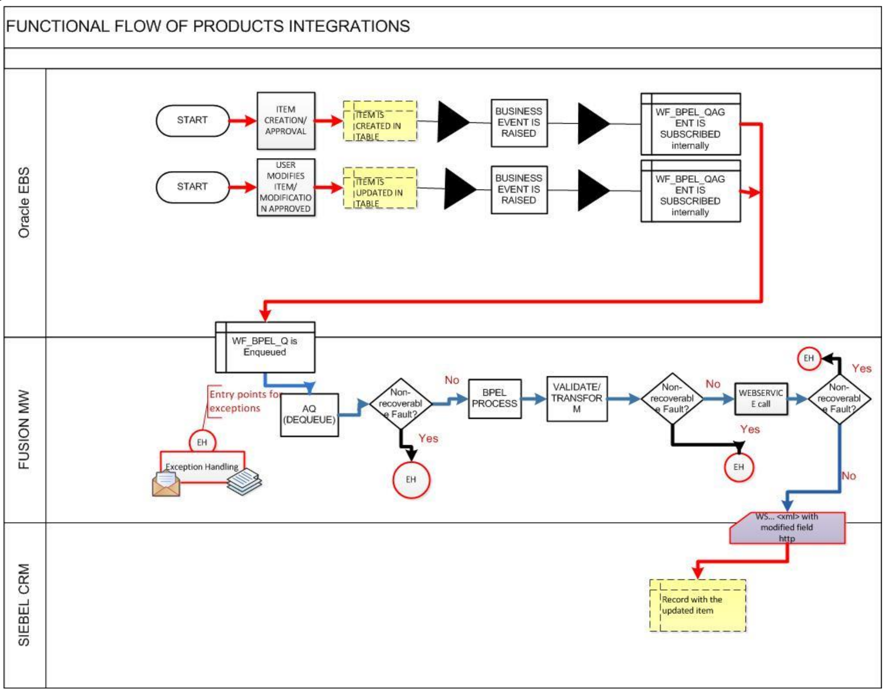
In our design, We are trying to integrate order to Insert or update an item/Item modification been approved from Oracle EBS to a simple product in Siebel CRM.Once a simple product is synchronized in Siebel, any changes to the product attributes or definition in Siebel CRM are not synchronized back to Oracle EBS.This process will not synchronize any structural details (e.g. from the EBS BOM structure) and will only synchronize item records from EBS into Siebel as simple products.Deleting items in Oracle EBS will not result in a request to delete simple products in Siebel CRM because no event mechanisms exist to accomplish this operation. Any deletions of simple products in Siebel CRM must be run as a manual process by an administrator in Siebel CRM.If an existing item name in Oracle EBS is updated and synchronized to Siebel, a product will be updated in Siebel.

METHODOLOGY

Key Assumptions (for Design Decisions)

- Oracle EBS is the source for Products data.
- We are trying to integrate in order to Insert or update an item/Item modification been approved from Oracle EBS to a simple product in Siebel CRM
- Once a simple product is synchronized in Siebel, any changes to the product attributes or definition in Siebel CRM are not synchronized back to Oracle EBS.
- This process will not synchronize any structural details (e.g. from the EBS BOM structure) and will only synchronize item records from EBS into Siebel as simple products.
- Deleting items in Oracle EBS will not result in a request to delete simple products in Siebel CRM because no event mechanisms exist to accomplish this operation. Any deletions of simple products in Siebel CRM must be run as a manual process by an administrator in Siebel CRM
- If an existing item name in Oracle EBS is updated and synchronized to Siebel, a product will be updated in Siebel.
- This process doesn’t cover Initial & Delta loads(conversions)

Process flow Diagram



Process flow description
This is a high-level description of the process flow for the Products interface from Oracle EBS to Siebel CRM. Custom SOA integration approach enables the synchronization (updating) of Products from Oracle EBS to simple products in Siebel CRM. Initiated by EBS, this one-way feed facilitates the updating of items (Integrations) from EBS to Siebel.

Design Overview

Interface Type	Real Time
Direction	EBS TO Siebel
Sessions /Tasks	
Schedule/Frequency	Real-Time Interface

Detailed Design Description

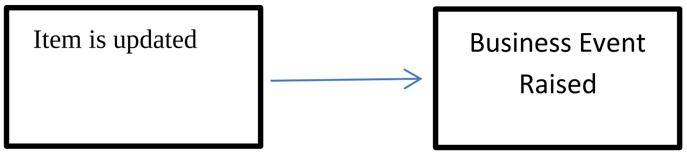
- **Requirements:**
  - When an item is created in EBS, the newly created product will flow through EIM and will be created in Siebel.
  - When an item is updated in EBS, the updated product will flow through the Fusion Middleware (Oracle SOA) and the corresponding product record will be updated in Siebel.
  - In each of the above scenarios – item creation and updates – this synchronization is a process initiated by the creation or update transactions in EBS.

Technical Design

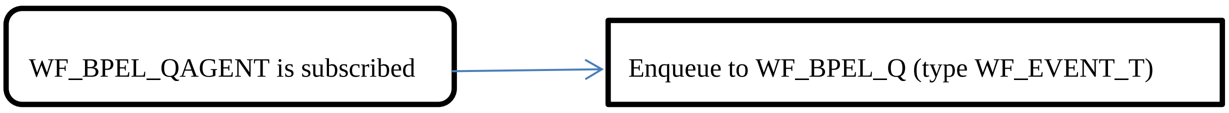
1. INBOUND (FROM EBS TO Fusion) (Using AQ Adapter)

1. We are going to leverage Oracle Workflow Business Event System (BES) that leverages the Oracle Advanced Queuing (AQ) infrastructure to communicate business events between systems.
2. When item is updated in EBS, Business event pertaining to product is raised. This event needs to be subscribed on EBS system by Integration repository administrators from Oracle Integration Repository user interface. Internally an event subscription is automatically created for that event with WF_BPEL_QAGENT as Out Agent and enqueued in WF_EVENT_T structure to Advanced Queue WF_BPEL_Q. A confirmation message appears if the event subscription is successfully created.
3. We will create a BPEL process to first dequeue the subscription from the WF_BPEL_Q queue to get the event details. The event details will be passed through BPEL process activities, properly transformed, validated and error handling done to the end system.
3. This event can be located from Oracle Integration repository either In-built with Oracle EBS suite or customized (patches should be applied properly)
4. If the BPEL process is successfully executed after deployment, we should get the same information from the output XML document once item is updated.

Initial Step



When Business Event is raised
Business Event **is subscribed on integration repository** and internally following takes place.


WF_BPEL_QAGENT is subscribed
Enqueue to WF_BPEL_Q (type WF_EVENT_T)

We need to dequeue from WF_BPEL_Q to get the event details using AQ adapter and pass it through BPEL to end system.
5. If a BPEL process is created with the business event that you have subscribed to it, in order for the subscribed business event to be successfully enqueued to WF_BPEL_Q queue, you need to make sure:

1. The consumer name must be unique.
2. The BPEL process is deployed before raising the business event.
Once the subscription is created and enqueued, the developer can then orchestrate the subscribed event into a meaningful business process in Oracle JDeveloper using BPEL language at design time.
EBS TABLES are provided
Tables and fields need to be mapped between EBS and Siebel CRM

2. BPEL PROCESS MANAGER WILL BE USED FOR PROCESS/SERVICE ORCHESTRATION

BPEL is the standard for assembling a set of discrete services into an end-to-end process flow, reducing the cost and complexity of process integration initiatives. Oracle BPEL Process Manager offers a comprehensive and easy-to-use infrastructure for creating, deploying and managing BPEL business processes. We would be defining one-way pattern (from EBS to Siebel) in BPEL

3. OUTBOUND From Fusion Middleware to Siebel CRM (WEB SERVICE)

1. We need to add a web service adapter to Reference Binding Component.
2. Web Services enables you to integrate with a standards-based web service using SOAP over HTTP and to create a web invocation service.
3. Web services are described in the WSDL file which is provided from the SIEBEL CRM side.
4.when a new service is added to a BPEL service component, the BPEL service component is notified to create a partner link that can be connected to a receive or an on-message activity.
5. When a service provided by a service component is deleted, all references to that service component are invalidated and the wires removed.

Please refer Integrations document which shows the tables and fields to be mapped between EBS and Siebel CRM

4. ERROR HANDLING

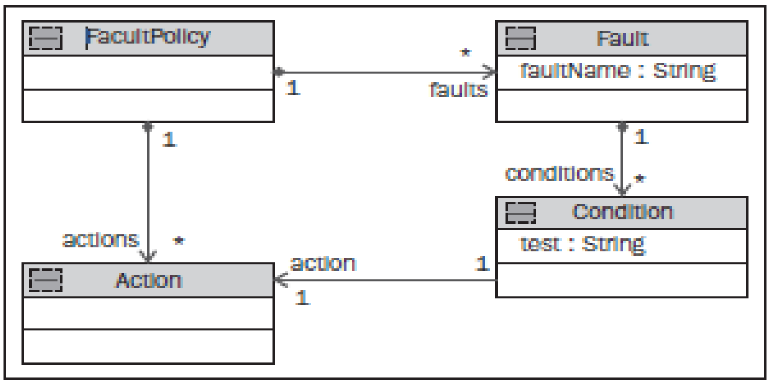
With service-oriented applications, the need for strong **exception handling** is even more important as the quality of service becomes a critical factor in **service reusability**. Exceptions that occur in a service-oriented are the **System faults** which are generally thrown by the underlying infrastructure when a runtime, system-level error occurs and the **Business faults** are thrown when an exception condition occurs during processing of a service operation, in the business logic of the service. This exception condition may be due to the violation of certain business rules, to invalid data, or to other non-system related conditions.
We will be incorporating the **POLICY-BASED FAULT HANDLING FRAMEWORK enabling Automated Fault handling**

Oracle SOA Suite 11*g* provides sophisticated error handling capabilities that enable you to easily define exception handling at various levels in a composite. It allows you to handle both system faults and application-generated business faults. XML. These exception-handling policies are attached to a composite and are not defined within its code. This allows you to reuse policy definitions across multiple components, composites, and projects. Fault-handling policies are defined in a file called **fault-policies.xml.**
A fault policy is defined by:

- Fault policy name (Fault Policy)
- A list of faults in a policy (Fault)
- A set of fault conditions (Condition)
- A set of fault recovery actions corresponding to the fault and the fault condition (Action)

The following class diagram shows the relationships between the elements in a fault policy.

DOI: 10.17148/IJARCCE.2025.14309



Since a fault policy definition is not specific to a composite, you need a way to attach appropriate policies to your composite. This is done by binding a fault policy to a composite or to one or more components within a composite. This binding definition is specified in **fault-bindings.xml**.

Sample Fault-Policy file:

```
            fault-policies.xml:
            <?xml version="1.0" encoding="UTF-8" ?>
            <faultPolicies xmlns="http://schemas.oracle.com/bpel/faultpolicy">
            <faultPolicy version="2.0.1"
            id="POProcessingFaults"
            xmlns:env="http://schemas.xmlsoap.org/ soap/envelope/"
            xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="http://schemas.oracle.com/bpel/faultpolicy"
            xmlns:xsi="http://www.w3.org/2001/ XMLSchema-instance">
            <Conditions>
            <!-- Step #1: Add your fault handler for remote fault here: -->
            <!-- Step #2: Add your fault handler for binding fault here: -->
            <!-- Step #3: Add your fault handler for mediator faults here: -->
            </Conditions>
            <Actions>
            <!-- Step #4: Add the Action definition for handling mediator faults using custom java here:-->
            <!-- Custom Java Handler: Logs the fault details to a log file -->
            <Action id="my-java-handler">
            <javaAction className="soatraining.faulthandling. MyFaultHandler"
            defaultAction="ora-terminate" propertySet="myProps">
            <returnValue value="OK" ref="ora-rethrow-fault"/>
            </javaAction>
            </Action>
            <!-- Retry -->
            <Action id="ora-retry">
            <retry>
            <retryCount>4</retryCount>
            <retryInterval>2</retryInterval>
            <exponentialBackoff/>
            </retry>
</Action>
            <!-- Rethrow action -->
            <Action id="ora-rethrow-fault">
            <rethrowFault/>
            </Action>
            <!-- Human Intervention -->
            <Action id="ora-human-intervention">
            <humanIntervention/>
            </Action>
            <!-- Terminate -->
            <Action id="ora-terminate">
            <abort/>
            </Action>
            </Actions>
            <!-- Property sets used by custom Java actions -->
            <Properties>
            <!-- Property set for MyFaultHandler customer java action -->
            <propertySet name="myProps">
            <property name="logFileName">myfaulthandler.log</property>
```

```
        <property name="logFileDir">c:\temp</property>
        </propertySet>
        <!-- Step #5: Add new property set for MyFaultHandler for logging Mediator faults here:-->
        </Properties>
        </faultPolicy>
</faultPolicies>
```

FAULT-BINDING FILE

```
        <?xml version="1.0" encoding="UTF-8" ?>
        <faultPolicyBindings version="2.0.1"
        xmlns="http://schemas.oracle.com/bpel/ faultpolicy"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<composite faultPolicy="MyFaults"/>
</faultPolicyBindings>
```

We can also define your own **custom exception handler using Java** wherein we can include any custom error handling methods like **EMAIL NOTIFICATIONS.**We can make use of the fault handling feature of BPEL to handle the binding faults returned by the DB adapter.Similarly we can make use fault handling feature of BPEL to handle the binding faults returned by Webservice call.We can even leverage the feature of email notifications from human task feature!We can even use reporting feature from worklist service component of Human tasks feature.

5. PROPER CONFIGURATIONS, TRANSFORMATIONS AND DEPLOYYMENT

All the components, services and references must be properly WIRED and proper TRANSFORMATIONs must be done to ensure the data flow.
After having properly configured, wired, transformed, validated, variable properties properly set(if needed),error handling done we should be ready to build and deploy the composite onto the server and then test it on server! Any change in EBS tables should be able to flow through fusion to the corresponding tables in the Siebel .Siebel Tables should be queried to check the changes.

Related Configuration Details
Configuration Data Field Requirements Reference

Siebel Base Table	Siebel Base Table Column	Siebel BC Name	Siebel Field Name
S_PROD_INT	X_SC_OM_GRP	SWI Internal Product VOD	SC OM Group
S_PROD_INT	X_SC_OM_SERIES	SWI Internal Product VOD	SC OM Series
S_PROD_INT	X_SC_OM_MATTRESS_STYLE	SWI Internal Product VOD	SC Mattress Style
S_PROD_INT	X_SC_OM_TYPE	SWI Internal Product VOD	SC OM Type
S_PROD_INT	X_SC_OM_SIZE	SWI Internal Product VOD	SC OM Size
S_PROD_INT	X_SC_OM_AIR_CONTROL	SWI Internal Product VOD	SC OM Air Ctrl
S_PROD_INT	X_SC_REASON_GROUP	SWI Internal Product VOD	SC Reason Group
S_PROD_INT	X_SC_FINANCE_GROUP	SWI Internal Product VOD	SC Finance Group
S_PROD_INT	X_SC_WARRANTY_GROUP	SWI Internal Product VOD	SC Warranty Group
S_PROD_INT	X_SC_LAYELG_FLG	SWI Internal Product VOD	SC Layaway Flag
S_PROD_INT	X_SC_ENG_DESC	SWI Internal Product VOD	SC Engineering Description
S_PROD_INT	X_SC_HD_FLG	SWI Internal Product VOD	SC HD Flag
S_PROD_INT	X_SC_FR_FLG	SWI Internal Product VOD	SC Fire Retd Flag
S_PROD_INT	X_SC_BOX_NUM	SWI Internal Product VOD	SC Box Size
S_PROD_INT	X_SC_RMA_CRI_CMP_FLG	SWI Internal Product VOD	SC RMA Flag

S_PROD_INT	X_SC_WEIGHT_UOM_CD	SWI Internal Product VOD	SC Weight Unit of Measure

List of Value Dependencies
EBS
1. Access EBS database credentials info needed(read/write access to test the changes)
2. Make sure Oracle server entries into your host file
3. Add the JNDI entry for DB to the adapter deployment descriptor (weblogic-ra.xml)
4. The list of adapter instances should be stored in a deployment descriptor file, weblogic-ra.xml on Oracle WebLogic Server. (It is inside of AQAdapter.rar, which contains also the Java class files in AQAdapter.jar).
5. Connection to the database instance by where E-Business suite is running: setting up the datasources, setting username/pwd, getting the correct JDBC driver, SID, Port and connection URL.
6. Database Schema name and tables information
7. Information about the Business Event to be used in "message selector rule"
6. **Subscription to business event from Oracle Integration repository user Interface must be done by Integration repository (or EBS) Admins**
8. Business Event must be available in Oracle Integration Repository to be subscribed to. Any kind of customizations for the business events must be taken care of.
9. If a BPEL process is created with the business event that you have subscribed to it, in order for the subscribed business event to be successfully enqueued to WF_BPEL_Q queue, you need to make sure:

 1. The consumer name must be unique.
 2. The BPEL process is deployed before raising the business event.

Once the subscription is created and enqueued, the developer can then orchestrate the subscribed event into a meaningful business process in Oracle JDeveloper using BPEL language at design time.

SIEBEL
1. WSDL
2. Siebel database credentials
3. Make sure SIEBEL server entries into your host files (tnsnames)

Record Reconciliation
Publishing Results: Siebel Web service completes the request and returns a response message. In the process, the cross-reference is linked to the Siebel IDs of the product.

For Unprocessed records refer "Handling of Unprocessed records" section.

Data Retention: simple products will not be deleted from Siebel application. This can be inactivated by setting End Dates to the product records.
Handling of Unprocessed records: For Siebel Web Service EAI Object Manger logs can be verified for errors. Once the issue is fixed whether it is data issue or code issue the unprocessed records will be re-published from Oracle EBS to Siebel CRM.
APPENDIX
Sample output XML format form Siebel.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <SOAP-ENV:Body>
    <ns:SWIProductImportUpsert_Output xmlns:ns="http://siebel.com/asi/V0">
      <ListOfProductImport xmlns="http://www.siebel.com/xml/SWIProductIntegrationIO">
        <ProductIntegration searchspec="?" operation="Upsert">
          <WorkspaceId/>
          <WorkspaceName>Workspace 09/11/2012 03:38:03</WorkspaceName>
          <ListOfProductDefinition>
            <ProductDefinition searchspec="?" operation="Upsert">
              <AutoExplodeFlag>N</AutoExplodeFlag>
              <BillableFlag>Y</BillableFlag>
              <BusinessUnitId>0-R9NH</BusinessUnitId>
              <CheckEligibilityFlag>N</CheckEligibilityFlag>
              <Description>TestFromPIP</Description>
              <EffectiveEndTime/>
              <EffectiveStartTime/>
              <InclusiveEligibilityFlag>N</InclusiveEligibilityFlag>
              <OrderableFlag>N</OrderableFlag>
              <PartNumber>SC1234</PartNumber>
              <PriceType>One-Time</PriceType>
              <ProductId>1-KB2R3</ProductId>
              <ProductName>SC1234</ProductName>
              <ReleaseFlag>Y</ReleaseFlag>
              <SalesProductFlag>Y</SalesProductFlag>
              <ServiceInstanceFlag>N</ServiceInstanceFlag>
              <ServiceProductFlag>N</ServiceProductFlag>
              <StructureType>Simple</StructureType>
              <TrackAsAssetFlag>Y</TrackAsAssetFlag>
              <Type>Product</Type>
              <UnitofMeasure>Case</UnitofMeasure>
              <VendorId/>
              <BillingServiceType/>
              <BillingType/>
              <ProductClassId/>
              <ProductClassName/>
              <ListOfSwiAdminProductLineVbc>
                <ProductLineVbc Searchspec="?" Operation="Upsert">
                  <ProductLineName>SOGNO</ProductLineName>
                </ProductLineVbc>
              </ListOfSwiAdminProductLineVbc>
            </ProductDefinition>
          </ListOfProductDefinition>
        </ProductIntegration>
      </ListOfProductImport>
    </ns:SWIProductImportUpsert_Output>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Migration Steps:

The following objects need to be migrated from DEV->TEST->PROD

Business Components

- SWI Internal Product VOD
- SWI Product Definition VBC

Integration Objects

- SWI Admin ISS Product Definition
- SWIProductIntegrationIO

Business Service

- SWI Product Import

Test/Execution Steps:

- When an item is created in EBS, the newly created product will flow through the SOA and will be created in Siebel.
- When an item is updated in EBS, the updated product will flow through the SOA and the corresponding product record will be updated in Siebel.
- In each of the above scenarios – item creation and updates – this synchronization is an automated process initiated by the creation or update transactions in EBS.
- Testing is done on Admin console to check if the flow is successful. Apart from this the SIEBEL Tables can be queried to check if the change is reflected from EBS to Siebel
- Unit Testing and Integration Testing needs to be done.

We can even leverage the built-in testing framework

Oracle SOA Suite 11g/12c provides a testing framework that allows you to:

- Define tests, assertions, and emulations using JDeveloper
- Run these tests either from the EM console or on the command line using ANT
- Review the test results from the EM console or as a JUnit report

There are three parts to a test case:

1. An Initiation of a composite defines the service and operation invoked along with the test data.
2. Emulation defines the message or fault returned from a reference or component invoked through a synchronous response or a callback without executing the component or referenced service.
3. An Assertion compares the message or part of the message over an SCA wire against the expected data.

A single test includes definitions of the initiation, emulations, and assertions. The test suite is defined in the composite project and is deployed along with the composite to the server, so it can be initiated from the EM console.

CONCLUSION

This design can be used for data flow from Oracle EBS 11i to Siebel CRM 8 leveraging Oracle SOA as middleware and for easy flow of data from Oracle EBS 11i to Siebel CRM 8 for the product interface. Basically this invention is done for manufacturing/consumer /retail businesses. It can work in any of these sectors. It can be used across different corporations leveraging Oracle SOA 11g as middleware and having data flow from oracle EBS 11i/R12 to Siebel CRM 8.The database used should be Oracle database and it could be clustered(RAC) or non-clustered. Also, it could produce easy flow of data between Oracle EBS 11i/R12 and Siebel CRM 8