

# How To Reconcile Customers Between Oracle eBS & Siebel CRM

Miroslav Samoilenko



January 2008

## How To Reconcile Customers Between Oracle eBS & Siebel CRM

In the world of best-of-breed applications, data integration has become a critical activity. Many Customers are choosing Siebel CRM to handle their front office operations and Oracle eBusiness Suite to handle their back office processing. Understanding the options to achieve seamless integration between the two applications is the key challenge for any implementation team.

In this article, I will consider the task of customer reconciliation between Oracle eBusiness Suite and Siebel CRM specifically the propagation of any changes to the customer record from Oracle eBusiness Suite to Siebel.

### The Old Dog: FTP

The first solution to consider is that of file exchange. Using this approach, a report is built which scans through the customer accounts table in Oracle eBusiness Suite and selects all records which have changed since the last run of the report. The outputs from this process are delivered in an XML file. A secondary process reads the output of this report and places it into a dedicated directory from which Siebel is able to read.

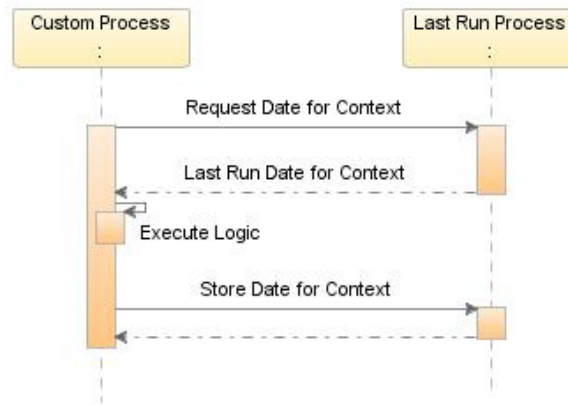
In order to make this approach work, several auxiliary processes need to be built:

#### Last Run Process

The Last Run Process is a utility which remembers the last date of execution of a particular concurrent program for a context. It is possible to use the existing history of execution of concurrent requests, but there are several shortcomings to this. First of all, the history gets periodically purged.

If the frequency of execution of the process and the purging frequency does not reconcile, you may never know when you process was last executed. The second issue is the context. For example, if the process is meant to retrieve data for the current operating unit, it may be rather difficult to determine the operating unit based on the parameters used for the process.

These two limitations often lead to the build of a custom process. Custom processes interact with the Last Run Process via two procedures. One procedure returns the last execution date for the given request and context. The other one stores the new execution date for the given request and context. Knowing the request, the Last Run Process determines the concurrent program which is being executed. The value for the context is defined by a developer, and may be either an operating unit, if the process retrieves data for a specific operating unit, or NULL.

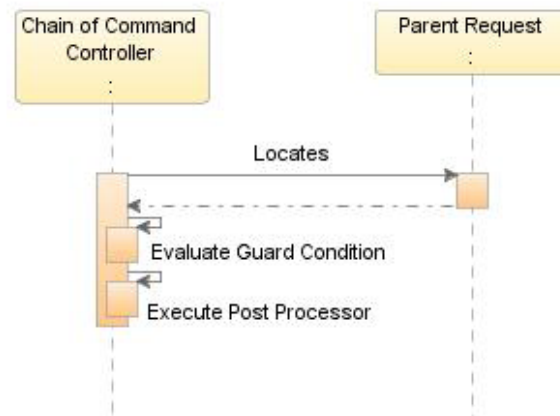


### Chain of Command Controller

It is always desirable to separate the data extraction process from the transportation process. In the case of the transportation process failing, it is possible to hold previously generated data until a successful process is achieved. Such separation allows us to build additional logic into the transportation process. For example, if we generate the report only for a given account, we need to email the output to the account manager, while if we extract data for all customers, we need to FTP it to Siebel.

The decision making about where and how the extracted data is processed can be offloaded to an implementation of Chain of Command design pattern. The necessity of such a pattern is clear - we need to process the output of a successfully finished concurrent process. We could have used request sets to generate data and then transport it. However, we would not be able to reuse it to email customer statements to the customer.

The implementation of the Chain of Command is simple. Concurrent programs are associated with a guard condition and a post-processor. The Chain of Command Controller concurrent program is scheduled to run periodically. It scans the concurrent programs which have successfully completed since the last run of the controller, locates the registrations for these programs, evaluates the guard condition which confirms if the post processor needs to be invoked and then executes the post processor.



The post processor is a PL/SQL procedure which can either execute another concurrent program, send an email using available API, or do any other action which may be needed.

Combination of the Last Run Process and the Chain of Command Controller delivers a simple, generic and intelligent solution for data transfer which can be used to simplify check and customer statement printing processes using BI Publisher. This would otherwise require human intervention.

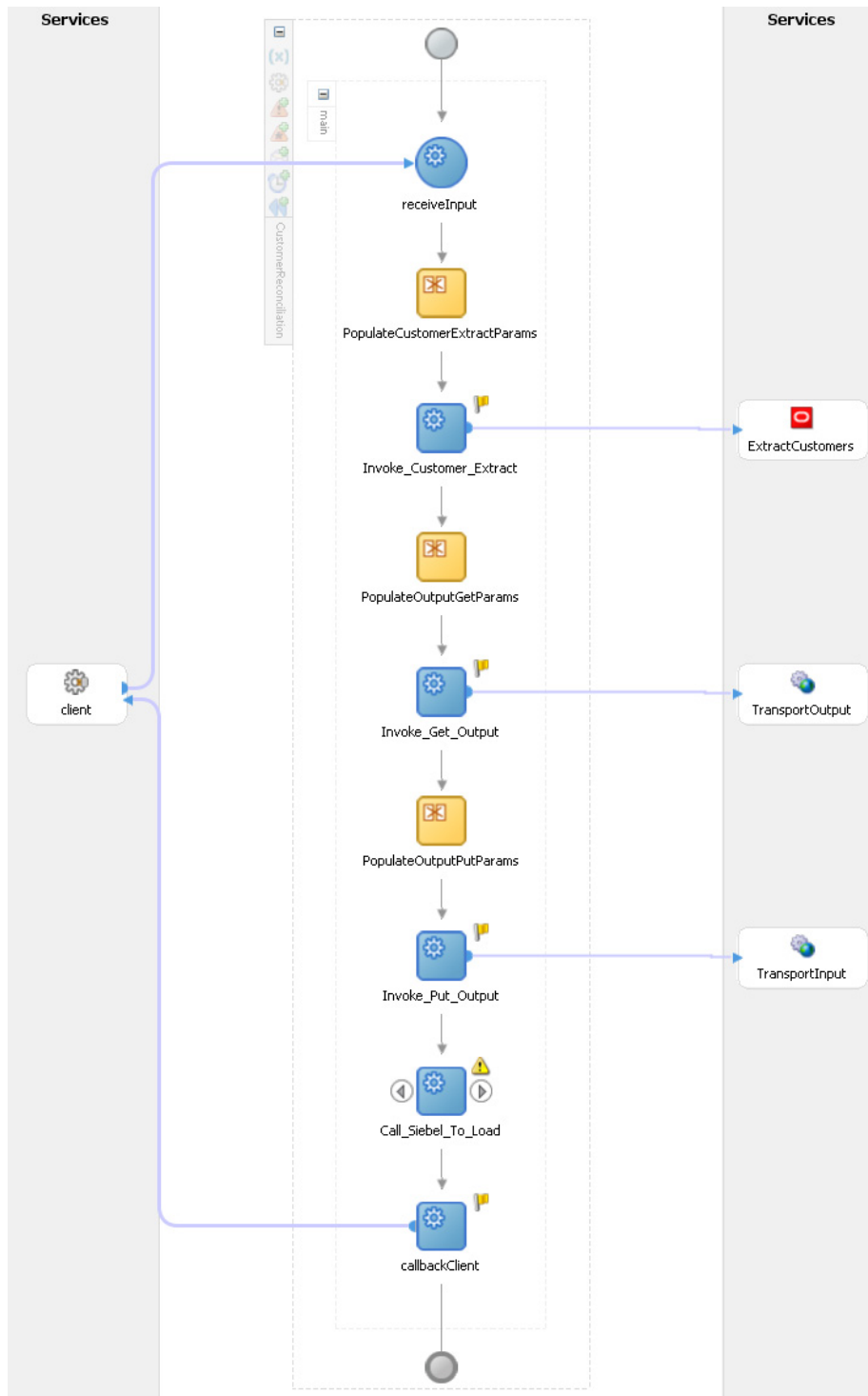
Error handling is a major challenge for this solution. It requires an individual to constantly monitor the chained processes for any failures. Automation of the error handling and notifications can make the solution prohibitively complex. This brings us to the point where we need a central control facility for the interfaces we build. Such central control is the BPEL Engine.

### **BPEL: Custom Solution**

Having built the data extract concurrent process, it can be used for the transportation to a BPEL Engine.

Oracle BPEL engine has multiple Adapters which allow you to execute concurrent programs in Oracle Applications, FTP files between locations, send messages to queue etc. This variety of Adapters significantly reduces your build time for custom BPEL processes. You can build very complex processes with complicated error handling mechanisms and compensations all from the comfort of JDeveloper 10g.

The process to FTP an output of a concurrent program to a remote file server can be represented by the following diagram:



In this BPEL process we use the Oracle Applications adapter to execute the concurrent program which extracts modified customer accounts. We then FTP its output to the BPEL server file system and then FTP it again to the Siebel Server file system. Once the file is delivered to Siebel, we call Oracle BPEL Siebel Adapter processes to upload the modified customers.

The diagram does not have any error handling or compensations. However, each invocation in the BPEL process can be surrounded by error handling procedures.

A BPEL engine is the solution of choice to build and monitor complex interfaces. The Oracle BPEL Engine provides a friendly UI to monitor execution of the processes and the Adapters. These standard components provide a great means to build user friendly processes.

### **Application Integration Architecture**

Application Integration Architecture is a set of tools, components and best practices which simplify integration between various applications. This is a concrete implementation of community knowledge about application integration.

Oracle AIA 1.0 provides the tools you need to integrate the Oracle eBusiness Suite, Siebel CRM, PeopleSoft, and J.D. Edwards with other applications. Industry best practices are incorporated into these process integration packs within Application Integration Architecture. One such integration pack is the Siebel CRM Integration Pack for Oracle Order Management.

This is a comprehensive solution which enables you to reconcile customers, items, assets, and orders between Oracle eBusiness Suite and Siebel CRM. The tool for this pack is Oracle BPEL Engine. It utilises the Oracle Application BPEL Adapter and Oracle Siebel Adapter to build a comprehensive BPEL Process which integrates these two applications.

The task of customer account reconciliation should firstly be based on the Siebel CRM Integration Pack for Oracle Order Management. This greatly reduces the implementation lifecycle, decreases the scope for custom development, and simplifies the task of interfacing between two applications used for configuring predefined processes.