

Oracle® Retail Invoice Matching
Operations Guide Addendum
Release 11.1.15

January 2009

Copyright © 2009, Oracle. All rights reserved.

Primary Author: Susan McKibbin

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software—Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

- (i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server – Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.
- (ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.
- (iii) the **SeeBeyond** component developed and licensed by Sun Microsystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.
- (iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.
- (v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.
- (vi) the software component known as **Access Via**TM licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.
- (vii) the software component known as **Adobe Flex**TM licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.
- (viii) the software component known as **Style Report**TM developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.
- (ix) the software component known as **DataBeacon**TM developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, “alteration” refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle’s licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Preface	vii
Related Documents.....	vii
Customer Support.....	vii
Review Patch Documentation	vii
Oracle Retail Documentation on the Oracle Technology Network.....	vii
Conventions.....	vii
1 Introduction	1
2 Technical Architecture	3
Overview	3
The Layering Model	4
Presentation Layer	4
Middle Tier	5
Data Access Layer (DAL).....	5
Database Layer	5
Technical Services	6
Third Party Libraries	7
ReIM-Related Java Terms and Standards.....	8
3 Batch Processes	11
Batch Architectural Overview	11
EDI-Related File-Based Batch Processes	11
Internal Batch Processes.....	12
Internal Batch Processes that Write to Staging Tables	12
Batch Processes that Extract from Merchandising System (RMS) Staging Tables	12
Batch Names	13
Functional Descriptions and Dependencies	14
Features of the Batch Processes.....	18
Scheduler and the Command Line	18
Batch Return Values	18
Batch Log and Error File Paths	18
Multi Threading Batch Processes.....	19
A Note about Restart and Recovery	19
Executing Batch Processes	20
Update ORACLE_HOME and REIM_HOME in the BatchRunner.sh file	20
Update the Script	20
Execute Batch Jobs Using Arguments from the Merchandising Batch Schedule.....	20
Terms Ranking Batch Design	21
Assumptions and Scheduling Notes	21
High-Level Flow Diagram.....	21
Primary Tables Involved.....	21

Batch Purge Batch Design.....	22
Assumptions and Scheduling Notes.....	23
Major Modules	23
Primary Tables Involved.....	23
Discrepancy Purge Batch Design.....	24
Major Modules	24
Major Tables	24
EDI Invoice Upload Batch Design	25
Assumptions and Scheduling Notes.....	25
Restart and Recovery.....	25
Primary Tables Involved.....	25
Auto-Match Batch Design.....	26
Algorithms.....	26
Assumptions and Scheduling Notes.....	27
Post Processing.....	27
High-Level Flow Diagram.....	28
Primary Tables Involved.....	28
Receipt Write-Off Batch Design	28
Assumptions and Scheduling Notes.....	29
High-Level Flow Diagram.....	29
Primary Tables Involved.....	30
Reason Code Action Rollup Batch Design	30
Assumptions and Scheduling Notes.....	31
High-Level Flow Diagram.....	31
Primary Tables Involved.....	32
Disputed Credit Memo Action Rollup Batch Design.....	32
Assumptions and Scheduling Notes.....	32
Primary Tables Involved.....	32
Financial Posting Batch Design.....	33
Assumptions and Scheduling Notes.....	33
Primary Tables Involved.....	34
Lookup Tables that must be Populated	34
EDI Invoice Download Batch Design.....	36
Assumptions and Scheduling Notes.....	36
Primary Tables Involved.....	36
Restart and Recovery.....	36
Complex Deal Upload Batch Design.....	36
Assumptions and Scheduling Notes.....	37
Primary Tables Involved.....	37
Fixed Deal Upload Batch Design	37
Assumptions and Scheduling Notes.....	37
Primary Tables Involved.....	38

Preface

Related Documents

For more information, see the following documents in the Oracle Retail Invoice Matching Release 11.0.15 documentation set:

- Oracle Retail Invoice Matching Batch Schedule
- Oracle Retail Invoice Matching Data Model
- Oracle Retail Invoice Matching Installation Guide
- Oracle Retail Invoice Matching Release Notes

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

<https://metalink.oracle.com>

When contacting Customer Support, please provide the following:

- Product version and program/module name.
- Functional and technical description of the problem (include business impact).
- Detailed step-by-step instructions to re-create.
- Exact error message received.
- Screen shots of each step you take.

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

Navigate: This is a navigate statement. It tells you how to get to the start of the procedure and ends with a screen shot of the starting point and the statement “the Window Name window opens.”

Note: This is a note. It is used to call out information that is important, but not necessarily part of the procedure.

This is a code sample
It is used to display examples of code

[A hyperlink appears like this.](#)

Introduction

The information in this document reflects modifications and updates to the latest ReIM Operations Guide. Using this document in conjunction with that guide provides retailers with a complete overview of the application.

Sections with changes are included in this document in full. For more specific information regarding modifications made to this Oracle Retail Invoice Matching release, see the Oracle Retail Invoice Matching 11.0.15 Release Notes.

Technical Architecture

This chapter describes the overall software architecture for ReIM. The chapter provides a high-level discussion of the general structure of the system, including the various layers of Java code.

Note that at the end of this chapter, a description of ReIM-related Java terms and standards is provided for your reference.

Overview

The system's Java architecture is built upon a layering model. That is, layers of the application communicate with one another through an established hierarchy and are able to communicate only with neighboring layers.

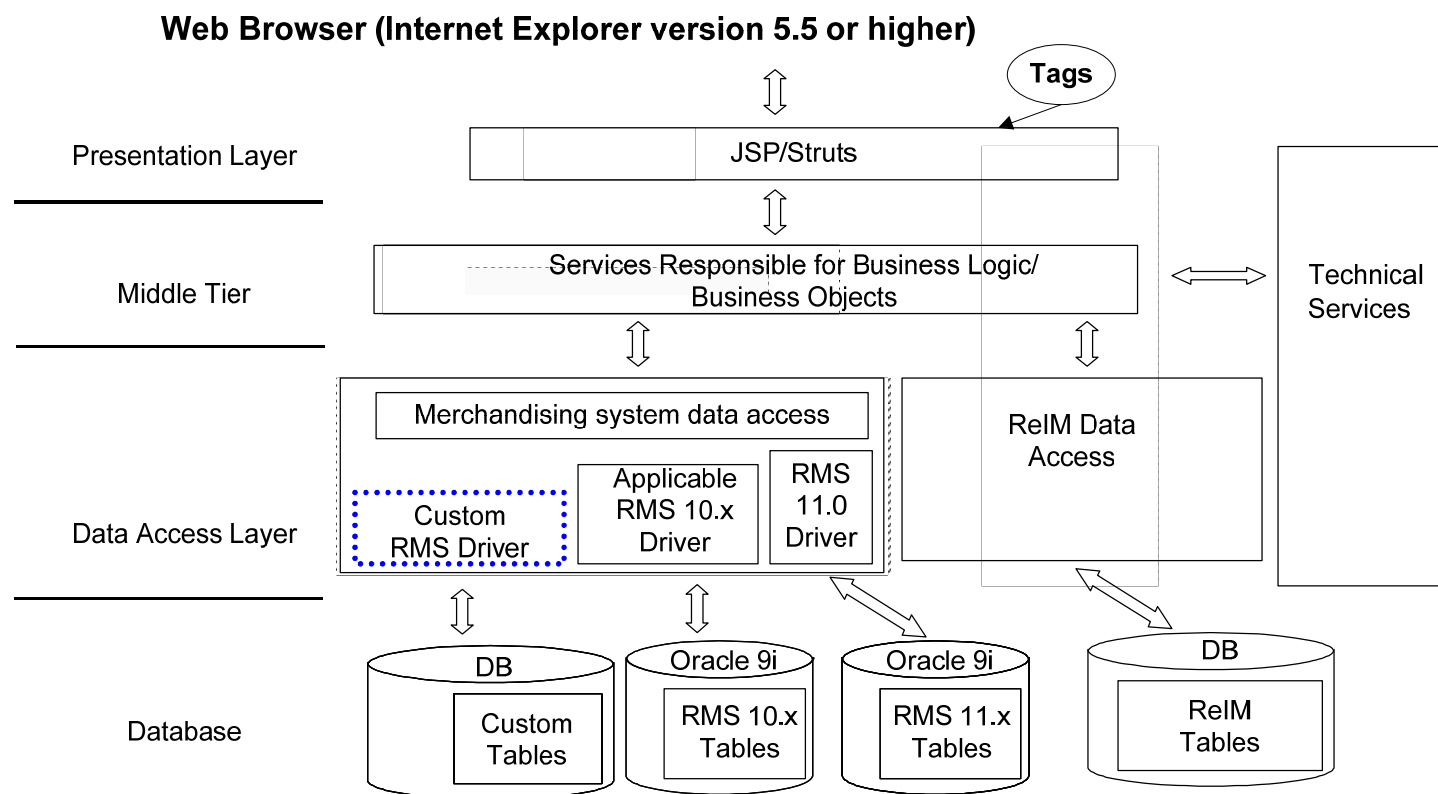
The application is divided into a presentation layer, a middle tier consisting of services and business objects, and a database access/driver layer. Technical services provide the "glue" that holds the application together, offering the application frameworks for error logging, internationalization, transaction management, application security, and so on.

The segregation of layers has the following advantages, among others:

- The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
- The look and feel of the application can be updated more easily because the GUI is not tightly coupled to the back end.
- A layered architecture has become an industry standard.
- Portions of the data access layer (DAL) can be radically changed without effecting business logic or user interface code.
- The application takes advantage of Java database connectivity (JDBC), minimizing the number of interface points that must be maintained.
- Market-proven and industry-standard technology is utilized (for example, JSPs, JDBC, and so on).

The Layering Model

The following diagram, together with the explanations that follow, offers a high-level conceptual view of the layers and their responsibilities within the architecture. Key areas of the diagram are described in more detail in the sections that follow.



ReIM Layered Architecture

Presentation Layer

This area of the architecture encapsulates the graphical user interface (GUI) processing. A web browser accesses JSP pages using a Struts tag library.

JSPs consist of JavaScript and standard HTML. They make calls to tag-libraries. An extension of Java servlet technology, JSPs are compiled into servlets. JSPs provide a user interface that can be separated from most of the business logic that resides on the server. This separation of presentation from content offers a greater possibility for ease of maintenance, both with regard to the page that the user sees and the underlying logic. The look and feel of the GUI is easy to customize, and dynamic functionality is easy to create.

Struts provide an open source framework for building Web applications. The core of Struts is a flexible control layer based upon Java servlets, JavaBeans, ResourceBundle, and Extensible Markup Language (XML). Struts provide an industry standard approach to enforcing the division between user interface code and business logic. Struts also provide standard functionality for error display, internationalization/screen translation, and so on. The Struts framework is part of the Jakarta Project, sponsored by the Apache Software Foundation (<http://www.apache.org/>). The official Struts home page is <http://jakarta.apache.org/struts>. The presentation layer only interacts with the middle tier services.

Middle Tier

Service Layer Responsible for Business Logic

The service layer consists of a collection of Java classes that implement business logic (data retrieval, updates, deletions, and so on) via one or more high-level methods. In other words, the service layer controls the workflow. For example, when a user clicks OK on a page, the server must follow a given series of steps to accomplish business functionality. The service layer controls how those steps are accomplished.

The service layer is the entry point to the middle tier and separates the presentation layer from the database layer. Generally the methods that are exposed by service layer classes accept and/or return business objects. The service layer encapsulates the business logic by calling down into business objects and the data access layer, thus making the code more maintainable.

Business Objects

Within ReIM, business objects are beans (that is, Java classes that have one or more attributes and corresponding set/get methods) that represent a functional entity. In other words, business objects can be thought of as data containers, which by themselves have almost no business functionality. (In those unusual cases where business logic resides within a business object, the logic pertains to a discreet business concept.) Two examples of business objects include Document and Supplier.

There is not necessarily a one-to-one relationship between a business object and a database table. The service layer may utilize more than one class from the data access layer in order to combine the data from more than one database table to fully populate a business object.

Data Access Layer (DAL)

The data access layer interacts only with the middle tier and the database. Classes in the DAL abstract the actual persistence mechanism that is being used to persist business objects. The DAL provides the mechanism that allows ReIM to be associated to a different persistence engine. Ideally, in those cases, only the DAL would need to be modified due to the change. The remainder of ReIM would continue to operate unchanged.

Database Layer

The database layer is the application's storage platform, containing the physical data (user and system) used throughout the application. This layer is only intended to deal with the storage and retrieval of information and is not involved in the manipulation of the data.

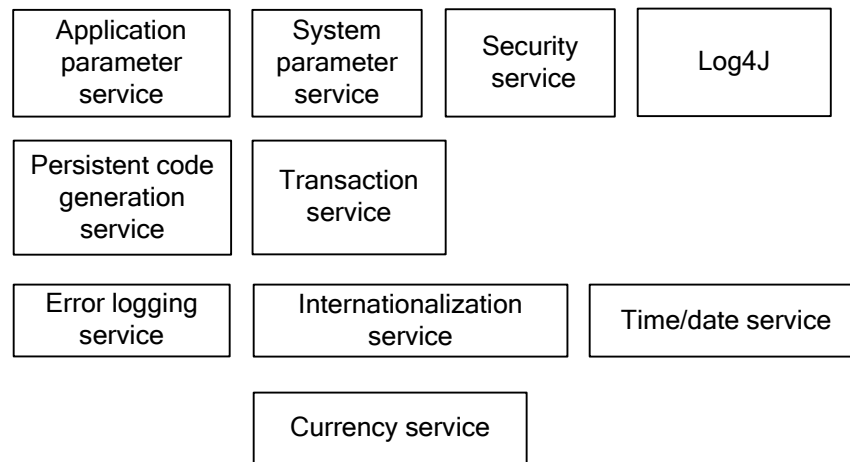
Technical Services

In order to increase the maintainability of the code, and enhance the rapid development of new business logic, a number of base technical services are provided.

Technical services hold the application together by providing common services to the application, services that are not necessarily driven by business requirements.

Technical services include application frameworks such as error logging, internationalization, transaction management, application security, and so on.

A brief description of each technical service follows the diagram.



ReIM Technical Services

Application Parameter Service

This service allows application configuration parameters to be stored within the database on a single database table. Developers can retrieve these parameters using a high level interface.

System Parameter Service

Similar to the application parameter service, this service is used only for technical configuration parameters. Although most configurable parameters are hosted in a system parameter table, some parameters are located in a properties file. See Chapter 2 – “Backend System Administration and Configuration” for more information.

Transaction Service

Note: The transaction service does not provide checkpoint transaction management or multi-phase commit.

This service provides a simplified management of rollback/commit semantics. In order to avoid the need to pass the database connection between the middle tier method calls and the data access layer classes, the transaction service uses thread local variables to maintain the current connection for a thread until that thread has committed or rolled back the transaction. This service thus simplifies transaction management.

Error Logging Service

This service incorporates a standard `ReIMException` class to raise and handle Java exceptions (shown below). The `ReIMException` class automatically logs itself to the application log file. The level of logging may be raised or lowered in the properties file. For example, an operator could configure the system to only display INFO and above. See Chapter 2 – “Backend System Administration and Configuration” for more information.

The system’s coding pattern ensures that the error messages, no matter where they originate, remain detailed in their presentation to the operator. For example, if a business specific message is thrown near the database, such as that an item-supplier combination does not exist, the system does not genericize that exception under a “could not post” transaction message or something similar. Rather, the error message is presented in all of its original detail for the operator.

Log4J

This service provides the error logging services with a standard method for logging information to a flat text file. Log4J is an open source product.

Internationalization Service

This service uses resource files to provide configurability for on-screen messages (such as on screen labels or error messages). To change the language for the ReIM GUI screens, a replacement set of resource files can be created. Note that although this service supports any number of languages, the screen flow remains left to right, top to bottom.

Currency Service

This service provides a high-level mechanism for developers to represent a currency amount. This service provides the formatted representation of that currency.

Time/Date Service

This service provides a high level interface to the Java time/date constructs along with some formatting methods for displaying these constructs on the GUI screens.

Security Service

The security service provides basic authorization and authentication functionality during user login. The association of the user to security roles controls user access to the functional areas of the application. The security service validates a user’s identity against a security store and retrieves the role memberships and role authorizations for that user upon a successful login. The physical implementation of the security information for each user, role, functional authorizations, and field authorizations is independently configurable among the database or LDAP server locations.

Third Party Libraries

ReIM base development uses the following third party libraries:

- Oracle JDBC library
- Log4J
- JUnit from www.junit.org
- Struts from jakarta.apache.org
- ICU4J from IBM
- Spring Framework from www.springframework.org

ReIM-Related Java Terms and Standards

ReIM is deployed using the technologies and versions described in this section.

The Java 2 Enterprise Edition (J2EE)

The Java standard infrastructure for developing and deploying multi-tier applications. Implementations of J2EE provide enterprise-level infrastructure tools that enable such important features as database access, client-server connectivity, distributed transaction management, and security.

Java Database Connection (JDBC)

JDBC is a means for Java-architected applications such as ReIM to execute SQL statements against an SQL-compliant database, such as Oracle. JDBC is part of Sun J2EE specification. Most database vendors implement this specification.

JDBC provides the support that allows ReIM to submit SQL queries to the database and receive the result set for further processing.

Java Development Kit (JDK)

Standard Java development tools from Sun Microsystems.

Java Server Pages (JSP)

JSPs enable Java and HTML to be combined within a web page. To the user, a JSP appears in the Web browser as a file with a .jsp extension. The JSP source is dynamically compiled into a servlet by the servlet container running in the web server. The servlet generates the necessary HTML content that the user sees.

Java Servlet

A servlet is a Java platform technology that allows a web application easier access to server side resources. The HTTP request from the client's browser is routed to the servlet, which then can process it as necessary and provide the applicable response to the user.

LOG4J

LOG4J is an open source sub-project of the Jakarta Project. It provides a configurable framework for logging information gathered during the execution of an application.

Naming Conventions in Java

- Packages: The prefix of a unique package name is written in all-lowercase letters.
- Classes: These descriptive names are unabbreviated nouns that have both lower and upper case letters. The first letter of each internal word is capitalized.
- Interfaces: These descriptive names are unabbreviated nouns that have both lower and upper case letters. The first letter of each internal word is capitalized.
- Methods: Methods begin with a lowercased verb. The first letter of each internal word is capitalized.

Struts

An open source web development framework from the Jakarta Project and sponsored by the Apache Foundation. The framework includes three major components:

- A controller servlet that dispatches requests to applicable ReIM Action classes.
- JSP custom tag libraries, and associated support in the controller servlet, that support ReIM in providing an interactive form-based application.
- Utility classes to support the following:
 - XML parsing
 - The automatic population of JavaBeans properties based on the Java reflection APIs
 - The internationalization of prompts and messages

Batch Processes

This chapter provides the following:

- An overview of the batch architecture
- A functional summary of each batch process, along with its dependencies
- A description of some of the features of the batch processes (batch return values, batch threading, and so on)
- Development designs for each batch process

Batch Architectural Overview

ReIM batch processes are run as Java applications. Batch processes engage in their own primary processing. However, they utilize services when they must engage in actions outside their primary processing (for example, when they utilize a helper method, touch the database, and so on).

Services retrieve the data on which the batch processes work to complete their tasks. As noted in “Chapter 3 – Technical Architecture,” the service layer consists of a collection of Java classes that implements business logic (data retrieval, updates, deletions, and so on) via one or more high-level methods.

The business logic occurs within the service code, while the technical processing occurs within the batch code.

Note the following characteristics of the ReIM batch processes:

- They are not accessible through a graphical user interface (GUI).
- They are scheduled by the retailer.
- They are designed to process large volumes of data.
- Although ReIM is a system that does not stop running, it is recommended that batch processes should be executed during periods when users are not in the system.

EDI-Related File-Based Batch Processes

ReIM EDI-related batch processes are file based. For example, they either input a flat file into the system (EDI invoice upload) from outside the system, or they output a flat file from the system (EDI invoice download) to be sent to another system (that of a vendor). Both the EDI invoice upload and the EDI invoice download batch processes are described later in this chapter.

Internal Batch Processes

Other batch processes within ReIM do not input or output files. Rather, the goal of these batch processes is to take a snapshot of potentially large amounts of data from the key tables within the database, transform that data through processing, and then return it.

Internal batch processes that are described later in this chapter include:

- AutoMatchBatch
- BatchPurgeBatch
- DiscrepancyPurgeBatch
- DisputedCreditMemoActionRollupBatch
- ReasonCodeActionRollupBatch

Internal Batch Processes that Write to Staging Tables

The third type of batch process within ReIM takes a snapshot of potentially large amounts of data from the key tables within the database, transforms that data through processing, and then writes that data to staging tables.

This communication process has been designed with the assumption that, during production, ReIM will reside within the same database as the merchandising system. Presumably, during implementation, the retailer will develop an optimum way to move the applicable data from the staging tables to the appropriate location for that data.

- The internal batch processes that write to staging tables are described later in this chapter.

Batch Processes that Extract from Merchandising System (RMS) Staging Tables

The fourth type of batch process within ReIM extracts data from merchandising system staging tables, create documents with the data, and write the data to ReIM tables. The batch processes that follow this processing pattern include the following:

- ComplexDealUploadBatch
- FixedDealUploadBatch

Batch Names

The following table describes ReIM batch processes. The table's order reflects the dependencies that exist among the ReIM batch processes but does **not** include any dependencies that exist between ReIM and the merchandising system it interacts with.

Batch name	Class (oracle.retail.reim.batch.jobs)
Terms ranking (This batch process is applicable only for those retailers using RMS 10.1 and earlier versions.)	TermsRankingService
Batch purge	BatchPurgeBatch
Discrepancy purge	DiscrepancyPurgeBatch
EDI invoice upload	EdiUploadBatch
Auto-match	AutoMatchBatch
Receipt write-off	ReceiptWriteOffBatch
Reason code action rollup	ReasonCodeActionRollupBatch
Disputed credit memo action rollup	DisputedCreditMemoResolutionRollupBatch
Financial posting	FinancialPostingBatch
EDI invoice download	EdiDownloadBatch
Complex deal upload	ComplexDealUploadBatch
Fixed deal upload	FixedDealUploadBatch

Functional Descriptions and Dependencies

The following table summarizes ReIM batch processes and includes both a description of each batch process's business functionality and its batch dependencies:

Batch Processes	Details	Batch Dependencies
Terms ranking (This batch process is applicable only for those retailers using RMS 10.1 and earlier versions.)	Retailers send terms ranking files to ReIM on a periodic basis (usually monthly). ReIM has built an API to read this file and populate the terms ranking table.	
Batch purge	This process deletes data from database tables while maintaining database integrity. This process deletes records from the ReIM application that meet certain business criteria (for example, records that are marked for deletion by the application user, records that linger in the system beyond certain number of days, and so on).	
Discrepancy purge	The discrepancy purging program deletes data from database tables while maintaining database integrity. This program deletes records from ReIM that have discrepancies of zero.	
EDI invoice upload	This batch process uploads merchandise, non-merchandise invoices, credit notes, debit memos, and credit note requests from the EDI into the invoice-matching tables.	
Auto-match	Auto-match is a system batch process that attempts to match invoices to receipts without manual intervention. Invoices that are in ready for match, unresolved, or multi-unresolved status are retrieved from the database to be run through the auto-match algorithm. The processing consists of three levels – summary, detail, and header (VAT only).	<ul style="list-style-type: none"> ▪ EDI upload (Invoice Matching) ▪ Receipt upload (Merchandising system, such as RMS)

Batch Processes	Details	Batch Dependencies
Receipt write-off	In order for retailers to track received goods not invoiced, they must have the ability to write-off these goods for financial tracking. ReIM has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the Receipt write-off process is run, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, the receipt is written off, and the invoice match status is closed.	Auto-match and any associated processing must run prior to this batch processing

Batch Processes	Details	Batch Dependencies
Reason code action rollup	<p>This batch process sweeps the action staging table and creates debit and credit memos as needed. Only a single debit or credit memo is created per invoice/discrepancy type, with line details from all related actions for the same discrepancy type. This process deletes these records when completed; they are deleted after posting. Note that a separate, retailer-created batch process sweeps the receiver adjustment table. The action staging table is used during posting to post the reason code actions to the financial staging table. A separate, retailer-created batch process sweeps the receiver adjustment table. The process compares the unit cost and/or quantity received for the item on the shipment with the expected unit cost and/or quantity on the IM_RECEIVER_COST_ADJUST and/or IM_RECEIVER_UNIT_ADJUST tables. If a match exists, the receiver cost and/or unit adjustment has occurred in RMS (or the equivalent merchandising system). As a result, the process sets the "pending adjustment" flag on IM_INVOICE_DETAIL table to false for the invoice line. The reason code actions are only rolled up for an invoice if no invoice lines on the invoice have any pending adjustments.</p>	

Batch Processes	Details	Batch Dependencies
Disputed credit memo action rollup	<p>The disputed credit memo action rollup process checks the records on the IM_REVERSAL_RESOLUTION_ACTION table and rolls up the credit memo detail lines by document/item/reason code. The rollup occurs only if all lines on a disputed credit memo have been completely resolved (that is, no cost or quantity discrepancy records remain for the credit memo).</p> <p>After the rollup, a new set of detail lines associated with the resolution reason codes replace the original set of detail lines associated with the debit reason codes on the IM_DOC_DETAIL_REASON_CODES table.</p>	The disputed credit memo action rollup must occur before resolution posting and after receiver adjustment.
Financial posting	<p>A recurring resolution posting process retrieves all matched invoices and approved documents.</p> <p>For each invoice, the batch process writes applicable financial accounting transactions to either of the following tables: IM_FINANCIALS_STAGE</p> <p>The AP staging tables, IM_AP_STAGE_HEADER and IM_AP_STAGE_DETAIL, if the RMS System-Options table: FINANCIAL-AP = O, Oracle-Financials-Vers = 1</p>	
EDI invoice download	<p>The EdiDownload module creates a flat file to match the EDI invoice download file format. The module retrieves all header, detail, and non-merchandise information and formats the data as needed.</p> <p>In other words, the EDI invoice download process retrieves debit memos, credit note requests, and credit memos in approved status from the resolution posting process and creates a flat file. The client converts the flat file into an EDI format by the client and sends it via the EDI invoice download transaction set.</p>	Auto-match must run prior to the EDI invoice download.

Batch Processes	Details	Batch Dependencies
Complex deal upload	This module reads data from RMS staging tables, creates credit memos, debit memos, and credit note requests out of the data, and stores the supporting deal data on a ReIM table for later use during posting.	The RMS staged data must be purged after the upload.
Fixed deal upload	This module reads data from RMS staging tables, creates credit memos, debit memos, and credit note requests out of those, and stores the supporting deal data on a ReIM table for later use during posting.	The RMS staged data must be purged after the upload.

Features of the Batch Processes

Scheduler and the Command Line

If the client uses a scheduler, batch process arguments are placed into the scheduler.

If the client does not use a scheduler, batch process parameters must be passed in at the UNIX command line.

Each of these scripts interacts with the generic shell script. These scripts take any and all arguments that their corresponding batch process would take when executing.

Batch Return Values

The following guidelines describe the batch process return values that ReIM batch processes utilize:

- SUCCESS = 0
- FAILED_INIT = 1
- FAILED_PROCESS = 2
- FAILED_WRAPUP = 3
- SUCCESS_WITH_REJECTS_TO_DB = 4
- SUCCESS_WITH_REJECTS_TO_FILE = 5
- SUCCESS_WITH_REJECTS_TO_DB_AND_FILE = 6
- UNKNOWN = -1

Batch Log and Error File Paths

Log file locations are determined by the retailer through the `reim.properties` file. If an error occurs that causes a batch process to suddenly come to a complete halt, the system writes to the error file. See Chapter 2 – “Backend System Administration and Configuration” for more information.

Multi Threading Batch Processes

The following batch processes shown below have multi-threading capabilities. The settings related to the multi-threading options for each batch process are established in the `reim.properties` file. See Chapter 2 – “Backend System Administration and Configuration” for more information.

Complex Deal Upload (ComplexDealUploadBatch)

This process is threaded by a group (or, bulk) of deals. Each group (or, bulk) constitutes a thread.

Fixed Deal Upload (FixedDealUploadBatch)

This process is threaded by a group (or, bulk) of deals. Each group (or, bulk) constitutes a thread.

EDI Invoice Upload (EdiUploadBatch)

This process is threaded by each transaction in the file (THEAD record to TTAIL record). Each thread handles transaction validation and insertion into the database (as valid or rejected) or facilitates the writing to a reject file.

Auto-Match (AutoMatchService)

Auto-match can either be run as a single thread or it can be threaded by the location hierarchy.

A Note about Restart and Recovery

Most ReIM batch processes do not utilize any type of restart and recovery procedures. Rather, if a restart is required, the process can simply be restarted, and it will start where it left off.

This solution is true for all batch processes other than those noted below:

- EDI invoice upload (its restart and recovery methods is described in its design below).
- EDI invoice download (its restart and recovery methods is described in its design below).

Executing Batch Processes

Formerly, batch processes were executed using the generic file. This generic file has been replaced by the BatchRunner.sh file for executing the batch processes. Retailers must manually update the REIM_HOME and ORACLE_HOME values in the BatchRunner.sh shell script file. Any custom script also must include the same changes to the REIM_HOME and ORACLE_HOME values.

Once the updates are made to BatchRunner.sh and to the custom script, batches can be run using the appropriate arguments for each job, as defined in the Merchandising Batch Schedule. For example, to run the ReIM batch process called AutoMatchBatch, the command is as follows (where username/password are parameters indicated by the Merchandising Batch Schedule):

```
BatchRunner.sh AutoMatchBatch username/password
```

Update ORACLE_HOME and REIM_HOME in the BatchRunner.sh file

The following are example values for ORACLE_HOME and REIM_HOME.

- ORACLE_HOME update
ORACLE_HOME=\$ORACLE_HOME
For example, ORACLE_HOME=/u03/webadmin/product/10.1.3/OracleAS_2
- REIM_HOME update
The war file is unpacked to this path:
REIM_HOME-\$ORACLE_HOME/j2ee/<instance-name>/applications/<context-root>/<app-name>
For example, REIM_HOME=\$ORACLE_HOME/j2ee/reim11-instance/applications/reim11/reim

Update the Script

If a script is used to execute batch processes, the script also must be updated as described below:

- The following is the program to be executed, as determined by the script name:
REIMPROG=`basename \$0`
REIMCLASS=" "
REIMCLASS2=" "
- Java home is expressed as: JAVA_HOME=
- For Java flags, use the following setting: JAVA_ARGS="-Xmx256M"
- Adjust the script as follows:
REIMCLASS=oracle.retail.reim.batch.BatchRunner

\$JAVA_HOME/bin/java -classpath \$REIMCLASSPATH \$JAVA_ARGS \$REIMCLASS "\$@"
- If there are two parts to the program, then:
if [\${REIMCLASS2}]; then
 \$JAVA_HOME/bin/java -classpath \$REIMCLASSPATH \$JAVA_ARGS \$REIMCLASS2
"\$@"
fi

Execute Batch Jobs Using Arguments from the Merchandising Batch Schedule

Refer to the Merchandising Batch Schedule for the arguments (or run parameters) required to execute each batch process.

Terms Ranking Batch Design

Note: This batch process is applicable only for those retailers using RMS 10.1 and earlier versions.

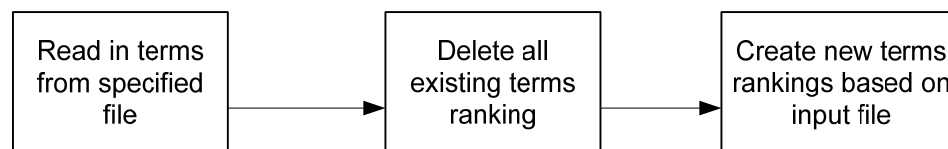
ReIM stores terms ranking information (from a financial system) on the terms ranking table. The information stored includes terms Ids and ranks. Oracle provides a terms ranking file format. Retailers are responsible for creating a file in the prescribed format. Retailers send terms ranking files to ReIM on a periodic basis (usually monthly). The ReIM system builds on API to read this file and populate the terms ranking table.

Additional terms information is obtained from the merchandising system (such as RMS). Specifically, Terms Description, Discount Days, and Percent data is read from the RMS Terms table. Terms Description (for example, 2.5%, 30 Days, Net Monthly, and so on) is displayed anywhere in the system where a particular Terms ID is displayed or edited. The Discount Days parameter is added to the Invoice Date to set the Due Date for an Invoice. The Percent parameter is used in an existence check. If a Percent value greater than 0 exists for a Terms ID, ReIM deduces that cash discounts apply to that Terms ID. This deduction is used wherever Cash Discount indicators are displayed.

Assumptions and Scheduling Notes

- The terms ranking interface program must be run as part of the batch cycle before the auto-match program. Methods from FinancialPostingService.java should be used to write to the financial staging table. If necessary, create new methods in this class to perform this processing.
- The program does not perform any validation to ensure that all terms are ranked, and so on. The program simply deletes all existing terms ranking, reads a file of terms and their ranking, and inserts the new term and ranking pairs into the ReIM database. Because the program deals with a relatively small number of records, the program does not have restart-recovery. If for some reason the program fails the entire load must be redone.
- This process should be run prior to auto-match; however, it will not run on a regular basis.

High-Level Flow Diagram



Primary Tables Involved

ReIM

- IM_TERMS_RANKING

RMS

- TERMS

Batch Purge Batch Design

The batch purging process (BatchPurge.java) deletes data from database tables while maintaining database integrity. This process deletes records from the ReIM application that meet certain business criteria (for example, records that are marked for deletion by the application user, records that linger in the system beyond certain number of days, and so on). The BatchPurge process does not generate any cascade relationships and/or SQL queries on the fly. The main features of the process are illustrated below:

Usage

The following arguments are applicable for the BatchPurgeBatch process:

```
BatchPurgeBatch userid/password PURGE [ALL|<table name>] [NOCOMMIT|COMMIT]
```

The first argument is a combination of user id and password. The second argument is the word PURGE. The third argument is either ALL or a single table name. Table name can be any one of the following:

1. IM_DOC_GROUP_LIST
2. IM_DOC_HEAD
3. IM_PARENT_INVOICE
4. IM_REASON_CODES
5. IM_PARTIALLY_MATCHED_RECEIPTS
6. IM_TOLERANCE_DEPT_AUDIT
7. IM_TOLERANCE_SUPP_AUDIT
8. IM_TOLERANCE_SUTRT_AUDIT
9. IM_TOLERANCE_SYS_AUDIT

ALL deletes data from all of the above tables. Finally, the fourth argument can be either NOCOMMIT or COMMIT. If there is no fourth argument, the default is NOCOMMIT.

SQL Queries

Delete statements have been optimized by minimizing the usage of nested SELECT statements and by maximizing the “table joins” in the WHERE clause. Any additions and/or modifications to the database require manual additions and/or modifications, respectively, to the existing SQL queries. All of the delete statements belonging to one cascade structure are added to a batch and executed at the end. It uses a single connection for each parent/children tree. Every cascade structure is a logical group.

Manual Propagation (Cascade) of Deletes to Child Tables

Every time there is a change in the relationship between tables, this process must be modified to reflect that change. Table relationship changes occur when clients decide to make significant customizations to the application.

Cascade Relationships

The developer must manually code the parent/child relationships between tables. For example, in order to delete records for the IM_DOC_HEAD table, records must be deleted from children tables in the following sequence of steps. Note that table sequence is not important within a single step.

Step 1

```
Delete from: IM_DETAIL_MATCH_INV_HISTOY
Delete from: IM_INVOICE_DETAIL_ALLOWANCE
Delete from: IM_QTY_DISCREPANCY_ROLE
Delete from: IM_QTY_DISCREPANCY_RECEIPT
```

Step 2

```
Delete from: IM_DOC_DETAIL_COMMENTS
Delete from: IM_MANUAL_GROUP_INVOICES
Delete from: IM_DOC_HEAD_COMMENTS
Delete from: IM_INVOICE_DETAIL
Delete from: IM_DOC_HEAD_LOCK
Delete from: IM_FINANCIALS_STAGE
Delete from: IM_COST_DISCREPANCY
Delete from: IM_RESOLUTION_ACTION
Delete from: IM_REVERSAL_RESOLUTION_ACTION
Delete from: IM_SUMMARY_MATCH_INV_HISTOY
Delete from: IM_QTY_DISCREPANCY
Delete from: IM_DOC_DETAIL_REASON_CODES
Delete from: IM_FINANCIALS_STAGE_ERROR
Delete from: IM_DOC_NON_MERCH
Delete from: IM_DOC_VAT
```

Step 3

```
Delete from: IM_DOC_HEAD
```

Cascade relationships are wired in the BatchPurge.java.

Assumptions and Scheduling Notes

Every time there is a change in the relationships among tables, the BatchPurge process has to be updated to accommodate these changes.

Major Modules**BatchPurge**

This class implements the batch delete process for the ReIM base application.

Primary Tables Involved

The following list includes the tables on which the purging algorithm is applied:

- IM_DOC_GROUP_LIST
- IM_DOC_HEAD
- IM_PARENT_INVOICE
- IM_REASON_CODES

Other tables of less significance also get purged.

Discrepancy Purge Batch Design

The discrepancy purging program deletes data from database tables while maintaining database integrity. This program deletes records from ReIM that have discrepancies of zero. Main features of the process are as follows:

- Usage

The following arguments are applicable for the DiscrepancyPurgeBatch process:

```
DiscrepancyPurgeBatch userid/password PURGE [ALL|<table name>]
```

Where the first argument is combination of user id and password. The second argument is the word PURGE. The third argument is either ALL or a single table name. Table name can be either of the following:

- IM_COST_DISCREPANCY
- IM_QTY_DISCREPANCY

ALL will delete data from all of the above-mentioned tables. Finally, the fourth argument can be either NOCOMMIT or COMMIT. If there is no fourth argument, the default will be NOCOMMIT.

- SQL Queries

The tables mentioned above are checked for merchandise invoices with cost and/or quantity discrepancies of zero. If they exist, the record is deleted from the table and the corresponding invoice detail line to will be updated to cost or qty matched. If the invoice line is now cost and qty matched the status of the line is set to matched and in return if all of the invoice lines are matched, the invoice itself is set to matched.

Major Modules

DiscrepancyPurge

Major Tables

- IM_COST_DISCREPANCY
- IM_QTY_DISCREPANCY
- IM_QTY_DISCREPANCY_RECEIPT
- IM_QTY_DISCREPANCY_ROLE
- IM_DOC_HEAD
- IM_INVOICE_DETAILS
- ORDSKU (RMS)
- ORDLOC (RMS)

EDI Invoice Upload Batch Design

EDI invoice upload is a standardized file format specification designed for vendors to send invoicing information electronically. The EDI invoice upload batch process performs the following:

- Reads each transaction within the file.
- Runs a file format validation (verifying file descriptors and line numbers; ensuring that numeric fields are all numeric and that character fields are all characters; looking for the invalid ordering of record type—THEAD followed directly by another THEAD; and so on). Certain file formatting errors cause the process to terminate with a message indicating the problem. A limited set of data validation errors cause the invalid transaction to be written to error tables (IM_EDIRECTORY_REJECT_DOC_XXX) where the data can be corrected through an online process. The rest of the data validation errors cause the invalid transaction to be written to a reject file where a user must correct the problems and re-run the file.
- Validates the data against the ReIM system and the merchandising system (such as RMS).
- Any errors found are recorded in an error log so that users can fix any transactions that were rejected to file.
- Adds the data to the ReIM system. All valid transactions are written to the IM_DOC_XXX, IM_INVOICE_XXX, IM_PARENT_XXX tables.

Assumptions and Scheduling Notes

- This process must be run before the auto-match process.
- All quantities are assumed to be in eases when uploaded.

Restart and Recovery

If the EDI invoice upload aborts without processing an entire file, the file needs to simply be rerun. When this action is completed, there will be multiple errors for the transactions that were successfully uploaded and the other transactions will be uploaded at that time as well. If the cause of the aborted process is software related, this fix may not solve the issue. Other steps may be required to ensure that the process completes its entire initial run.

Primary Tables Involved

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_INVOICE_DETAIL_ALLOWANCE
- IM_DOC_NON_MERCH
- IM_DOC_DETAIL_REASON_CODES
- IM_PARENT_INVOICE
- IM_PARENT_INVOICE_DETAIL
- IM_PARENT_NON_MERCH
- IM_EDIRECTORY_REJECT_DOC_DETAIL
- IM_EDIRECTORY_REJECT_DOC_DETAIL_ALLOW

- IM_EDI_REJECT_DOC_HEAD
- IM_EDI_REJECT_DOC_NON_MERCH
- IM_DOC_VAT

Auto-Match Batch Design

Auto-match is a system batch process that attempts to match invoices to receipts without manual intervention. Invoices that are in ready for match, unresolved, or multi-unresolved status are retrieved from the database to be run through the auto-match algorithm.

The three inputs into the auto-match process include the following:

1. Invoices
2. Receipts
3. Purchase orders

ReIM “owns” invoices, while receipts and purchase orders are owned by a merchandising system, such as RMS.

The processing consists of three levels: summary, detail, and header. Summary-level matching attempts to match all invoices to receipts at a summary level. Detail-level matching attempts to match all invoices (that do not match at a summary level) to receipts at a line item level. Header level matching attempts to validate VAT before continuing to attempt to match all invoices.

The auto-match process attempts to match the invoices to receipts to the best of its abilities. The process assigns different statuses according to the level of matching achieved.

If an invoice arrives prior to a receipt (for a particular PO), the auto-match process attempts only to match invoice unit cost to PO unit cost.

When a complete match cannot be made, manual intervention is required through online processes.

Algorithms

The following algorithms comprise the auto-match process:

1. Cost pre-matching
This process identifies any cost discrepancies prior to the arrival of receipts. If no receipts exist for the PO location, the invoices are sent to the cost pre-matching algorithm. Cost pre-matching is where unit costs on the invoice are compared with unit costs on the purchase order at a line level. If a match can be obtained, the invoice remains in ready-for-match status and is retrieved again for matching once the receipt comes in. If no match can be obtained, a cost discrepancy is created and routed immediately.
2. Summary matching
Invoices are grouped with receipts based upon purchase order location. A match is attempted for all invoices and receipts for the PO location. The invoices’ total extended costs are summed and compared with the receipts’ total extended costs. Based on a supplier option, the invoices’ total quantity is summed and compared with the receipts’ summed total quantity. If a match is achieved, all invoices and receipts are set to matched status. Otherwise, one-to-one matching is attempted for the PO location.

3. One-to-one invoice matching

This processing attempts to match a single invoice to a single receipt for the applicable PO location. If all invoices and receipts are set to matched status, the next PO location is processed.

If a multi-unresolved scenario exists (where more than one invoice can be matched with one or more receipts), all un-matched invoices are given the multi-unresolved status and no further processing occurs for this PO location.

4. Detail matching

During detail matching processing, an attempt is made to match each line on the invoice to an unmatched receipt line for the same item. Both the unit cost and quantity are always compared at the line level. If both the cost and quantity match, the invoice line and receipt line are placed into matched status. If the cost fails or the quantity fails, the cost or quantity discrepancies are generated and routed.

5. Header matching – (VAT only)

Invoices created without details are not able to have their VAT information validated at invoice creation. All header level only invoices are created with a status of Ready for Match. For VAT validation, this processing determines whether a header level only invoice that has been matched to a receipt should continue in the matching and posting process or whether it should be marked as having a VAT discrepancy and removed from the matching process.

Assumptions and Scheduling Notes

- Although not recommended, auto-match can be run during the day when there are users online interacting with the system.

Both the invoice unit cost and the POs unit cost must be expressed in the same currency. In order to compare the invoice unit costs with the POs unit costs, auto-match does not engage in currency conversion.

The system assumes that tolerance costs are always in the system's primary currency. If RMS is the applicable merchandising system, auto-match performs currency conversion if the currency on the order is different from the primary currency. RMS existing currency conversion engine is used to perform this conversion. If RMS is not being utilized, another currency conversion engine must be provided to support this functionality.

- The quantities on the invoice must be expressed in the same unit of measure as the quantities on the receipt. Auto-match performs no unit of measure conversion.
- The batch process runs after EDI upload (Invoice Matching) and Receipt upload (Merchandising system, such as RMS).
- Supplier options

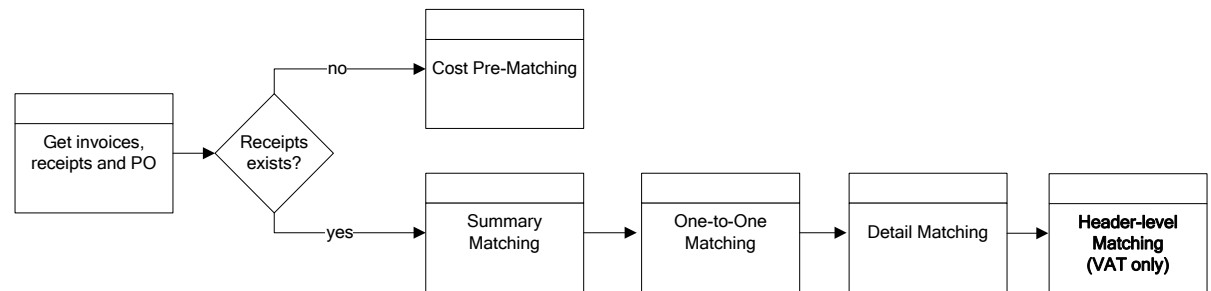
All suppliers must have options defined in order for their invoices to be processed by the system, and the terms defined for those suppliers have to be completely updated in RMS. In order to support the use of suppliers in ReIM, the ENABLED_FLAG (set to Y), START_DATE_ACTIVE and END_DATE_ACTIVE are the required entries in the TERMS_DETAIL table in RMS.

Post Processing

- Auto-match automatically invokes the best terms calculation for invoices that it matches.
- Auto-match automatically posts invoices that it matches.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the auto-match batch process.



ReIM Auto-Match Flow

Primary Tables Involved

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- SHIPMENT (RMS)
- SHIPSKU (RMS)
- IM_PARTIALLY_MATCHED_RECEIPTS
- ORDHEAD (RMS)
- ORDSKU (RMS)
- ORDLOC (RMS)
- IM_TOLERANCE_DEPT
- IM_TOLERANCE_SUPP
- IM_TOLERANCE_SYSTEM
- IM_COST_DISCREPANCY
- IM_QTY_DISCREPANCY
- IM_QTY_DISCREPANCY_RECEIPT
- IM_QTY_DISCREPANCY_ROLE
- IM_SUPPLIER_OPTIONS
- IM_SYSTEM_OPTIONS

Receipt Write-Off Batch Design

Retailers track received goods that are not invoiced, and they must have the ability to write-off these goods for financial tracking. Two types of processes can determine when these written-off goods will be written to financials: purged receipts from merchandising system, and “close open receipts” from invoice matching. Because receipts can be purged outside of the invoice matching dialogue, these purged receipts must be maintained until their unmatched amount has been accounted for. These receipts are tracked through STAGE_PURGED_SHIPMENTS and STAGE_PURGED_SHIPSKUS.

Every purged shipment record that is not fully matched will have a record by item written to the stage tables. In addition, invoice matching has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the write-off process is ran, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, then the receipt will be written off, and the invoice match status is closed.

The department/class of each receipt item must be identified to ensure accurate accounting. The form of the accounting distribution is as follows:

Transaction Type	Sign	Value	Notes
Unmatched receipt	Debit	Value of unmatched items on receipt	
Receipt write-Off	Credit	Same as above	
Trade accounts payable	Credit	0	Written as a matter of form

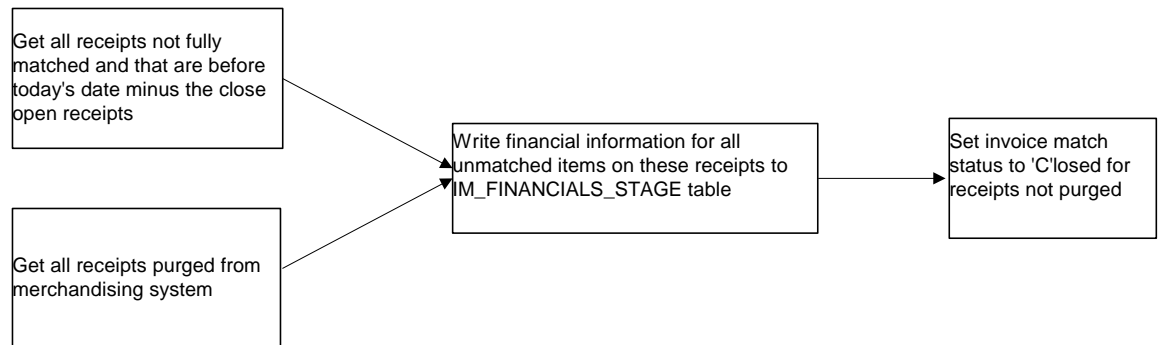
This account distribution mapping is set up through the account cross-reference screen.

Note: If IM_SUPPLIER_OPTIONS.CLOSE_OPEN_RECEIPT_MONTHS is not defined, the value is retrieved from IM_SYSTEM_OPTIONS.CLOSE_OPEN_RECEIPT_MONTHS.

Assumptions and Scheduling Notes

- When setting up the Close Open Receipt Months in ReIM Supplier Options and/or System Options, the value should be less than or equal to RMS UNIT_OPTIONS.ORDER_HISTORY_MONTHS if the intention is to have invoice matching pick up receipts prior to purging.
- Auto-match and any associated processing must be run prior to this batch processing.

High-Level Flow Diagram



Primary Tables Involved

ReIM

- IM_FINANCIALS_STAGE
- IM_SYSTEM_OPTIONS
- IM_SUPPLIER_OPTIONS
- IM_PARTIALLY_MATCHED_RECEIPTS

RMS

- UNIT_OPTIONS
- SHIPMENT
- STAGE_PURGED_SHIPMENT
- SHIPSKU
- STAGE_PURGED_SHIPSKU

Reason Code Action Rollup Batch Design

Reason code actions are resolutions assigned at the discrepancy line level. A number of fixed actions are available to resolve a line item discrepancy; the specific results depend on the action.

The resolution posting process sweeps the IM_RESOLUTION_ACTION table and creates debit and credit memos as needed. Only a single debit or credit memo is created per invoice/discrepancy type, with line details from all related actions for the same discrepancy type.

This process does not delete these records when completed; rather, they are deleted after posting.

A separate, client-created batch process sweeps the receiver adjustment table. The action staging table is used during posting to post the reason code actions to the financial staging table.

To resolve a cost discrepancy, the user can select a Receiver Cost Adjustment action from the cost resolution screen. Similarly, to resolve a quantity discrepancy, the user can select a Receiver Unit Adjustment action from the quantity resolution screen. The actions are written to the IM_RESOLUTION_ACTION table in an unrolled status with the amount of adjustment. The IM_INVOICE_DETAIL table also receives a flag that signifies “pending adjustment” for the invoice line.

At the same time, the actions are written to the IM_RECEIVER_COST_ADJUST and IM_RECEIVER_QTY_ADJUST tables to indicate the expected receiver adjustment amount on the RMS (or equivalent merchandising system) side. In sum, these two tables serve as the staging tables for the RMS (or equivalent merchandising system) process to actually perform the adjustment.

For a receiver cost adjustment, IM_RECEIVER_COST_ADJUST holds the order unit cost for the item after the adjustment. For a receiver unit adjustment, IM_RECEIVER_UNIT_ADJUST holds the received quantity for the item on the shipment after the adjustment.

The process compares the unit cost and/or quantity received for the item on the shipment with the expected unit cost and/or quantity on the IM_RECEIVER_COST_ADJUST and/or IM_RECEIVER_UNIT_ADJUST tables. If a match exists, the receiver cost and/or unit adjustment has occurred in RMS (or the equivalent merchandising system). As a result, the process sets the “pending adjustment” flag on IM_INVOICE_DETAIL table to false for the invoice line. The reason code actions are only rolled up for an invoice if no invoice lines on the invoice have any pending adjustments.

Because ReIM cannot control when and how the receiver adjustments are happening on the RMS side (or the equivalent merchandising system), records written to the IM_RECEIVER_COST_ADJUST and IM_RECEIVER_UNIT_ADJUST tables are considered final.

As a result, when the user resolves a cost or quantity discrepancy, the receiver adjustment must fully resolve a discrepancy before the user leaves the screen, and there should be no re-route actions involved. On the RMS side, the amount of adjustment must be exactly the same as expected.

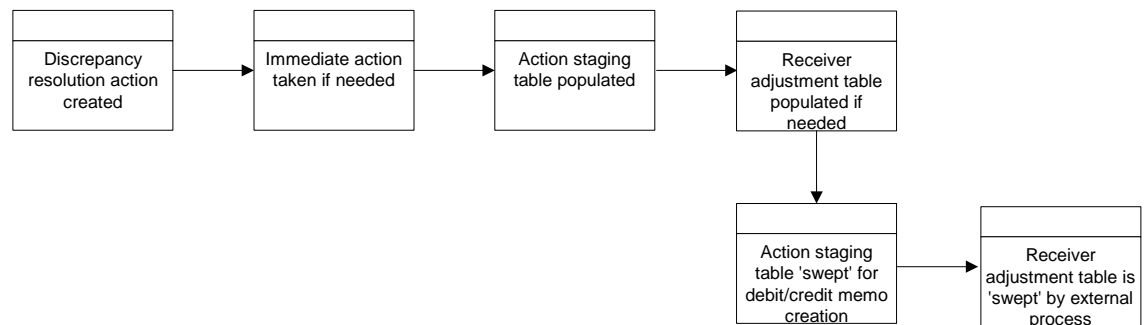
The IM_PARTIALLY_MATCHED_RECEIPTS table holds the amount of a receipt item that has been matched during invoice matching. The quantity received on the SHIPSKU table subtracts the quantity matched on the IM_PARTIALLY_MATCHED_RECEIPT table, giving the available to match quantity for the receipt item. Auto-match, summary matching, detail matching and quantity discrepancy resolution processes all keep track of the matched quantity bucket to determine how much of the receipt item has already been matched and how much of the receipt item remains available to be matched. In the case of a Receiver Unit Adjustment, the IM_PARTIALLY_MATCHED_RECEIPTS table is updated to reserve the entire remaining unmatched bucket for the receipt item. This logic prevents the adjusted receipt quantity from being used for any other matching or quantity resolutions.

Assumptions and Scheduling Notes

The memo staging table sweep must occur before the posting batch process, or a delay of one day results before posting can occur.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the reason code action rollup batch process.



ReIM Reason Code Action Rollup Flow

Primary Tables Involved

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_PARTIALLY_MATCHED_RECEIPTS
- IM_RESOLUTION_ACTION
- IM_RECEIVER_COST_ADJUST
- IM_RECEIVER_UNIT_ADJUST

Disputed Credit Memo Action Rollup Batch Design

When a disputed credit memo is first created as a reversal to a debit memo, cost, or quantity discrepancies are generated for each line on the credit memo, and the original debit memo reason codes are associated with the new credit memo detail lines.

As the user takes actions to resolve the discrepancy online, a record is written to the IM_REVERSAL_RESOLUTION_ACTION table for each resolution action taken. The only actions allowed to resolve the discrepancy are Deny Dispute or Approve Credit in Disputed Status. However, the user can choose multiple reason codes associated with Deny or Approve actions to resolve the disputed line. Also, the user can either resolve the disputed line completely, or partially resolve it. Upon complete resolution of a disputed line, the cost or quantity discrepancy is deleted from the system.

The disputed credit memo action rollup process checks the records on the IM_REVERSAL_RESOLUTION_ACTION table and rolls up the credit memo detail lines by document/item/reason code. The rollup occurs only if all lines on a disputed credit memo have been completely resolved (that is, no cost or quantity discrepancy records remain for the credit memo).

After the rollup, a new set of detail lines associated with the resolution reason codes replace the original set of detail lines associated with the debit reason codes on the IM_DOC_DETAIL_REASON_CODES table. The new credit memo lines are in Approved or Denied status depending on the resolution action. The credit memo header status is updated to Approved status. The lines that are approved are rolled up to calculate the header level total cost and total quantity. Non-merchandise costs can be associated with a credit memo that is created as a debit memo reversal, but no resolution actions can be taken on non-merchandise costs. Non-merchandise costs should be included in the credit memo total cost.

Assumptions and Scheduling Notes

The disputed credit memo action rollup must occur before resolution posting and after receiver adjustment.

Primary Tables Involved

The following tables are used for the debit memo reversal, resolution, and rollup processes:

- IM_DOC_HEAD

This table holds the document header information.

- **IM_DOC_DETAIL_REASON_CODES**
This table holds the document detail information by item/reason code. Before resolution rollup, this table holds the document detail information based on the original debit reason codes. After resolution rollup, this table holds the document detail information based on the reason codes used to resolve the disputed credit memo lines.
- **IM_REVERSAL_RESOLUTION_ACTION**
This table holds the resolution actions the user takes to approve or deny the disputed credit memo line.
- **IM_COST_DISCREPANCY**
This table holds the disputed credit memo lines for a debit memo cost reversal.
- **IM_QTY_DISCREPANCY**
This table holds the disputed credit memo lines for a debit memo quantity reversal.
- **IM_QTY_DISCREPANCY_ROLE**
This table holds the routing information for a credit memo quantity.

Financial Posting Batch Design

For each invoice, the batch process writes applicable financial accounting transactions to either of the following tables:

- The Financials staging table, IM_FINANCIALS_STAGE
- The AP staging tables, IM_AP_STAGE_HEADER and IM_AP_STAGE_DETAIL, or the IM_FINANCIALS_STAGE, depending on the transaction type (if the RMS System-Options table: Financial-AP = O, Oracle-Financials-Vers = 1)

The processing occurs after discrepancies for documents have been resolved by resolution documents. Once all of the resolution documents for a matched invoice are built, and all of the RCA/RUA external processing has been confirmed, the process inserts financial accounting transactions to the financials staging table, to represent the resolution and consequent posting of the invoice. The process also inserts financial accounting transactions for the approved documents that are being handled.

Once all of the transactions have been written, the process switches the status of the current invoices/documents to Posted, and then moves on to the next invoice/document.

If a segment look-up fails, the failed record is written to a financials error table.

Assumptions and Scheduling Notes

Before posting can occur, the following information must be set up:

- Set up segment definitions in the system.properties.
- Define GL account segments on the GL Options screen.
- Specify all the accounts using the GL Cross Reference screen.

The dynamic segments for a GL account can be any or all of the following designated segments:

- Country
- Location
- Dept
- Class

If dynamic segments are defined, the values for the segments must be defined in the applicable tables, IM_DYNAMIC_SEGMENT_DEPT_CLASS or IM_DYNAMIC_SEGMENT_LOC.

Primary Tables Involved

- IM_DOC_HEAD
Holds the matched and approved documents.
- IM_DOC_NON_MERCH
Holds the non-merchandise costs for invoices.

Lookup Tables that must be Populated

- IM_GL_OPTIONS
Order of segments and dynamic segments defined.
- IM_GL_CROSS_REF
Account values defined for account types and account codes.
- IM_DYNAMIC_SEGMENT_DEPT_CLASS
Accounts defined for each department/class combination.
- IM_DYNAMIC_SEGMENT_LOC
Accounts defined for each location/company combination.

Table to which the Process Posts Data

- IM_FINANCIALS_STAGE
 - Transaction code
 - Debit/credit indicator
 - Invoice ID
 - Invoice date
 - Supplier
 - Purchase order (if available)
 - Shipment/receipt (only if an unmatched receipt record is being written)
 - Currency
 - Amount
 - Best terms ID
 - Terms date
 - Pre-paid indicator
 - Comments
 - Create user ID
 - Create date-time
 - Segments that determine the mapping account in the external financial system (as defined in the IM_GL_CROSS_REF table).

OR

IM_AP_STAGE_HEAD

- Sequence Number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID; for identification purpose.
- Doc_id: Similar to IM_FINANCIALS_STAGE.

- Invoice Type Lookup Code: If document type = MRCHI or NMRCHI, this value is set to STANDARD. Otherwise this value is set to CREDIT.
- invoice_number: The concatenated data is as follows:
 - chars 1-34: the first 34 characters from the EXT DOC ID
 - char 35: a hyphen
 - chars 36-50: the DOC ID
- Vendor: Same as for current IM staging table.
- Oracle_site_id:
 - The loc from this transaction to read new RMS Location/Org Unit data to find the Org Unit.
 - The Org Unit to read new RMS Supplier Addr/Org Unit/Site ID data to find Oracle Site ID.
- Currency Code: Valued if this is a foreign currency invoice, otherwise null.
- Exchange Rate: If exchange rate is valued, this should be the literal USER; otherwise blank.
- Exchange Rate Type:
- Document Date: Same as in current IM staging table.
- Amount: The TOTAL amount including tax.
- Best Terms Date: Same as in current IM staging table.
- Segment1: Same as in current IM financials staging table.
- Segment2: Same as in current IM financials staging table.
- Segment3: Same as in current IM financials staging table.
- Segment4: Same as in current IM financials staging table.
- Segment5: Same as in current IM financials staging table.
- Segment6: Same as in current IM financials staging table.
- Segment7: Same as in current IM financials staging table.
- Segment8: Same as in current IM financials staging table.
- Segment9: Same as in current IM financials staging table.
- Segment10: Same as in current IM financials staging table.
- Create Date: Same as in current IM financials staging table.
- Best Terms ID: Same as in current IM financials staging table.

IM_AP_STAGE_DETAIL

- Doc_id
- Sequence number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID; for identification purpose.
- Transaction Code
- Line Type Lookup Code: This value varies. The rules are:
 - If the tran-code is UNR or VWT or REASON or CRN then this value is ITEM.
 - If this is a generated tax line, then this value will be TAX.
 - If none of the above, then this value will be MISCELLANEOUS.
- Amount
- Vat Code: Same as in current IM staging table. EXCEPT - for generated tax lines, the amount for this line should be the amount from the taxable line times the tax rate

- Segment1: For regular lines: same as in current staging table. For generated tax line: use values from source line.
- Segment2: (see rules for segment 1)
- Segment3: (see rules for segment 1)
- Segment4: (see rules for segment 1)
- Segment5: (see rules for segment 1)
- Segment6: (see rules for segment 1)
- Segment7: (see rules for segment 1)
- Segment8: (see rules for segment 1)
- Segment9: (see rules for segment 1)
- Segment10: (see rules for segment 1)
- Create Date: Same as in current IM staging table.

EDI Invoice Download Batch Design

The EDI invoice download process retrieves debit memos, credit note requests, and credit memos in approved or posted status from the resolution posting process and creates a flat file. The client converts the flat file into an EDI format and sends it via the EDI invoice download transaction set to the respective vendors.

Assumptions and Scheduling Notes

- All data is valid in the IM_DOC_HEAD tables. ReIM does not validate details.
- Auto-match must run prior to the EDI invoice download.

Primary Tables Involved

The EDI invoice download batch process reads from the following tables:

- IM_DOC_HEAD
- IM_DOC_DETAIL_REASON_CODES
- IM_DOC_NON_MERCH
- IM_DOC_DETAIL_COMMENTS

Restart and Recovery

If the EDI invoice download aborts while processing, an incomplete file is generated. To generate a complete file, the process simply needs to be rerun and allowed to fully process. If the cause of the aborted process is software related, this action might not solve the issue; other steps may be required to ensure that the process completes its entire initial run.

Complex Deal Upload Batch Design

The Complex Deal Upload batch process reads data from header and detail complex deals staging tables in RMS.

For each combination of deal ID and deal detail ID on the RMS staging tables, the batch process creates a credit memo, a debit memo, or a credit note request, depending upon an indicator on the staging tables.

The batch process also copies most of the data from the RMS staging tables into one ReIM detail table (IM_COMPLEX_DEAL_DETAIL). This data is later referenced during the posting process for the created documents.

Assumptions and Scheduling Notes

The RMS staging header and detail must be purged nightly after the upload has run.

Primary Tables Involved

Note: For descriptions of RMS tables, see the latest RMS data model.

- STAGE_COMPLEX_DEAL_HEAD (RMS table)
- STAGE_COMPLEX_DEAL_DETAIL (RMS table)
- IM_DOC_HEAD
This table holds general information for documents of all types. Documents include merchandise invoices, non-merchandise invoices, consignment invoices, credit notes, credit note requests, credit memos, and debit memos. Documents remain on this table for SYSTEM_OPTIONS.DOC_HISTORY_MONTHS after they are posted to the ledger.
- IM_DOC_DETAIL_REASON_CODES
This table contains quantity/unit cost adjustments for a given document/item/reason code.
- IM_DOC_VAT
This table associates the document with its value added tax (VAT) information.
- IM_COMPLEX_DEAL_DETAIL
This table holds the details of the complex deal stored in ReIM. It is used during complex deal detail posting.

Fixed Deal Upload Batch Design

The Fixed Deal Upload batch process reads data from header and detail fixed deals staging tables in RMS.

For each deal ID on the RMS staging tables, the batch process creates a credit memo, a debit memo, or a credit note request, depending upon an indicator on the staging tables.

The batch process also copies most of the data from the RMS staging tables into one ReIM detail table (IM_FIXED_DEAL_DETAIL). This data is later referenced during the posting process for the created documents.

Assumptions and Scheduling Notes

The RMS staging header and detail must be purged nightly after the upload has run.

Primary Tables Involved

Note: For descriptions of RMS tables, see the latest RMS data model.

- STAGE_FIXED_DEAL_HEAD (RMS table)
- STAGE_FIXED_DEAL_DETAIL (RMS table)
- IM_DOC_HEAD

This table holds general information for documents of all types. Documents include merchandise invoices, non-merchandise invoices, consignment invoices, credit notes, credit note requests, credit memos, and debit memos. Documents remain on this table for SYSTEM_OPTIONS.DOC_HISTORY_MONTHS after they are posted to the ledger.
- IM_DOC_NON_MERCH

This table holds various user-defined non-merchandise costs associated with an invoice. Non merchandise costs can be associated with merchandise invoice if the IM_SUPPLIER_OPTIONS.MIX_MERCH_NON_MERCH_IND for the vendor is Y. If the MIX_MERCH_NON_MERCH_IND for the vendor is N, non merchandise expenses can only be on non merchandise invoice documents.
- IM_DOC_VAT

This table associates the document with its value added tax (VAT) information.
- IM_FIXED_DEAL_DETAIL

This table holds the details of the fixed deals in the ReIM system. It will be used during fixed deal detail posting.