Oracle® Invoice Matching Cloud Service

Operations Guide

Release 16.0.21

E87300-01

May 2017

ORACLE®

Oracle Retail Invoice Matching Cloud Service Operations Guide, Release 16.0.21

E87300-01

Primary Author: Nathan Young

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

Send Us Your Comments ix

Preface xi

Audience xi
Documentation Accessibility xi
Customer Support xi
Review Patch Documentation xii
Improved Process for Oracle Retail Documentation Corrections xii
Oracle Retail Documentation on the Oracle Technology Network xii
Conventions xii

1 Introduction

What is Retail Invoice Matching? 1-1
Oracle Retail-Based Enterprises 1-2

2 Backend System Administration and Configuration

System Assumptions 2-1
reim.properties File 2-2
Logging Configuration 2-2
Log4J Conventions 2-3
Log4J Properties 2-3
Internationalization 2-3
Translation 2-4
Setting the User Language 2-4
Setting Date, Time, and Number Formats 2-5
Translations 2-5
ReIMResources.properties 2-5
batch.properties 2-5

3 Integration

Integration Overview 3-1
From the Supplier (to EDI) to ReIM 3-2
From ReIM (to EDI) to the Supplier 3-2
From ReIM to the Staging Table for Financial Systems Interface 3-2

From the Merchandising System to ReIM (Directly and Through EDI) ... 3-2
From ReIM to Receiver Unit and Cost Staging Tables to RMS ... 3-3
From ReIM to the Merchandising System ... 3-3
Electronic Data Interchange (EDI) Tables and Files ... 3-4
The EDI Reject Table ... 3-4
The EDI Reject File ... 3-5
EDI Injector File Layout (Based on EDI 810) ... 3-5
All Files Layouts Input and Output ... 3-5
Notes ... 3-18
EDI Invoice Download File Layout (Based on EDI 812) ... 3-19
Document Induction via UI ... 3-24
Financial System Interface ... 3-25
Foundation Financial Data Overview ... 3-25
Location Account Segments ... 3-25
Department/Class Account Segments ... 3-25
Financial Transactions ... 3-25
Complex and Fixed Deal-Related Posting ... 3-26
Financial Posting ... 3-26
Tracking Receipt Posts ... 3-26
Tables Related to Tracking Receipt Posts ... 3-26
Multiple Lines for an Individual Receipt Item ... 3-27
LDAP and Other User Interfaces ... 3-27
LDAP ... 3-27
Setup Steps within LDAP ... 3-28
Setup Steps within ReIM ... 3-28
Additional LDAP Resources ... 3-29

4 Technical Design

Locking Design Summary ... 4-1
Currency Design Summary ... 4-2
Merchandising System (such as RMS) and ReIM Assumptions ... 4-2
Currency Conversion Process for Amount Tolerances ... 4-2
Currency-Related System Validations ... 4-3
Currency Formatting ... 4-3

5 System Configuration

Define System Options ... 5-1
Define Supplier Options ... 5-2
Define Matching Tolerance ... 5-2
Define matching Strategies ... 5-2
Define Reason Codes ... 5-2
Define GL Mappings ... 5-4

6 Batch Processes

Batch Architectural Overview ... 6-1
Batch Process Configuration ... 6-2

EDI-Related File-Based Batch Processes 6-2
Internal Batch Processes 6-3
Internal Batch Processes that Write to Staging Tables 6-3
Batch Processes that Extract from Merchandising System (RMS) Staging Tables 6-3
Batch Names 6-3
Functional Descriptions and Dependencies 6-4
Features of the Batch Processes 6-6
Scheduler and the Command Line 6-6
Batch Return Values 6-6
Batch Log and Error File Paths 6-6
Multi-Threading Batch Processes 6-7
Complex Deal Upload (ComplexDealUploadBatch) 6-7
Fixed Deal Upload (FixedDealUploadBatch) 6-7
EDI Injector (EdiInjectorBatch) 6-7
Auto-Match (AutoMatchBatch) 6-7
A Note about Restart and Recovery 6-7
Executing Batch Processes 6-7
Tables Purge Batch Design 6-8
Usage 6-8
Purge Operational 6-8
Purge Workspace 6-8
Purge Workspace and Operational 6-8
Primary Tables Involved 6-8
Operational 6-8
Workspace 6-9
Accounts Purge Batch Design 6-9
Usage 6-9
Major Modules 6-9
Major Tables 6-9
EDI Invoice Injector Batch Design 6-9
Usage 6-10
Assumptions and Scheduling Notes 6-10
Restart and Recovery 6-10
High-Level Flow Diagram 6-10
Primary Tables Involved 6-10
Invoice Auto-Match Batch Design 6-11
Usage 6-12
Algorithms 6-12
Assumptions and Scheduling Notes 6-15
High-Level Flow Diagram 6-15
Primary Tables Involved 6-16
Credit Note Auto-Match Batch Design 6-17
Usage 6-18
Algorithms 6-18
Assumptions and Scheduling Notes 6-19
Post Processing 6-19
High-Level Flow Diagram 6-20

Primary Tables Involved 6-20
Receipt Write-Off Batch Design 6-22
Usage 6-23
Assumptions and Scheduling Notes 6-23
High-Level Flow Diagram 6-23
Primary Tables Involved 6-23
REIM 6-23
RMS 6-23
Reason Code Action Rollup Batch Design 6-23
Usage 6-24
Assumptions and Scheduling Notes 6-24
High-Level Flow Diagram 6-24
Primary Tables Involved 6-24
Financial Posting Batch Design 6-24
Usage 6-25
Assumptions and Scheduling Notes 6-25
Primary Tables Involved 6-25
Lookup Tables that must be Populated 6-25
Tables to Which the Process Posts Data 6-26
Financial System Integration 6-28
EDI Invoice Download Batch Design 6-29
Usage 6-29
Assumptions and Scheduling Notes 6-29
Primary Tables Involved 6-29
Restart and Recovery 6-29
Complex Deal Upload Batch Design 6-29
Usage 6-30
Primary Tables Involved 6-30
Multi-Threading 6-30
BlockSize 6-30
PartitionNo 6-30
Generation of Debit Memo (or Credit Note Requests) for Deals 6-31
Fixed Deal Upload Batch Design 6-31
Usage 6-31
Primary Tables Involved 6-32
Multi-Threading 6-32
BlockSize 6-32
PartitionNo 6-32
Generation of Debit Memo (or Credit Note Requests) for Deals 6-33

Send Us Your Comments

Oracle Retail Invoice Matching Cloud Service Operations Guide, Release 16.0.21

Oracle welcomes customer comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the new Applications Release Online Documentation CD available on My Oracle Support and www.oracle.com. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at www.oracle.com.

Preface

This Operations Guide provides critical information about the processing and operating details of Product, including the following:

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementers
- Business analysts who need information about Product processes and interfaces

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Customer Support

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

https://support.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

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.1). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

To more quickly address critical corrections to Oracle Retail documentation content, Oracle Retail documentation may be republished whenever a critical correction is needed. For critical corrections, the republication of an Oracle Retail document may at times not be attached to a numbered software release; instead, the Oracle Retail document will simply be replaced on the Oracle Technology Network Web site, or, in the case of Data Models, to the applicable My Oracle Support Documentation container where they reside.

This process will prevent delays in making critical corrections available to customers. For the customer, it means that before you begin installation, you must verify that you have the most recent version of the Oracle Retail documentation set. Oracle Retail documentation is available on the Oracle Technology Network at the following URL:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

An updated version of the applicable Oracle Retail document is indicated by Oracle part number, as well as print date (month and year). An updated version uses the same part number, with a higher-numbered suffix. For example, part number E123456-02 is an updated version of a document with part number E123456-01.

If a more recent version of a document is available, that version supersedes all previous versions.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

http://www.oracle.com/technetwork/documentation/oracle-retail-100266.html

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1
Introduction

Oracle Retail Invoice Matching (ReIM) provides a critical control function to verify invoices against corresponding merchandise purchase receipts prior to payment of the supplier invoice. ReIM naturally complements the Oracle Retail Merchandising System (RMS), which supports ordering, receiving, and other inventory management functions in the purchasing cycle.

ReIM accurately and efficiently verifies supplier invoices against corresponding receipt data. When total invoice cost and quantity is supported by one or more receipts (that is, the quantity received in the system, valued at the negotiated purchase order cost) within pre-defined tolerances, the invoice is verified (or matched) and is ready for payment. Where differences exist between invoice and receipt, a dialog supports the resolution process. Invoices with resolved discrepancies can be paid. Invoices verified for payment are staged in a table for a retailer to extract to their accounts payable and general ledger solutions.

ReIM is designed as a standalone application, with logic built in to reference any merchandising system. However, integration between ReIM and RMS is very robust and offers a compelling business case to the retailer.

Note: The Merchandising Cloud Services are based on corresponding on-premises applications. References to the on-premises application names exist throughout the Merchandise Cloud Services applications and documents.

What is Retail Invoice Matching?

Invoice matching describes a control procedure designed to ensure the retailer pays the negotiated cost for actual quantities received. Invoice verification or matching is a fundamental and critical control procedure for every retailer.

ReIM is designed to support the invoice verification process with accuracy and efficiency, focusing resources on exception management. ReIM accepts electronic invoice data uploads (EDI), and provides for rapid on-line summary entry of invoices. ReIM supports automated and on-line processes allowing one or more invoices to be matched against one or more receipts. When an invoice cost and quantities are matched within tolerance, it is ready for payment and staged to a table to allow a retailer to extract to their accounts payable solution.

If a cost or quantity difference between the invoice and receipts is outside tolerance, a discrepancy is recognized and must be resolved. A flexible resolution process allows discrepancies to be directed to the most appropriate user group for disposition. Reviewers are empowered to assign one or more reason codes that they are authorized to use, to resolve the discrepancy.

Each reason code is associated to a type of action (for example, create charge back or receiver cost adjustment). Many reason codes may be associated with a particular action type, allowing for more granular reporting, and so on. Actions drive document creation and EDI downloads to suppliers, inventory adjustments, and accounting activities. Actions also allow the invoice to be extracted by the retailer and posted for payment.

ReIM is highly integrated with RMS to drive efficiency, lower maintenance costs and improve control. ReIM integration provides access to the following data and more:

- RMS foundation data (organizational and merchandising hierarchies, supplier data, currency, exchange rates, and so on)
- Receipts tables and receiver adjustments
- Self-billing transactions (consignment purchases, direct store deliveries, and so on)
- RTV billings
- Deals and rebate bill-backs

Other functionality within ReIM supports credit note matching against credit note requests (issued in resolution of invoice discrepancies, as well as for RTVs and so on), supplier-disputed debit memos, best terms and terms date processing, flexible tolerance definition dialog, and so on.

Oracle Retail-Based Enterprises

Although ReIM has been developed as a stand-alone product, the most efficient implementation would be as part of the Oracle Retail product suite. This integration provides the following important benefits:

- The number of interface points that need to be maintained is minimized.
- The amount of redundant data and processes within the retail organization is limited.
- Future enhancements allow for greater extensibility into the retail enterprise.
- Delays in product introductions can be minimized.

2

Backend System Administration and Configuration

This chapter of the operations guide is intended for administrators who provide support and monitor the running system.

The content in this chapter is not procedural, but is meant to provide descriptive overviews of the key system parameters that establish the ReIM environment.

See the *Oracle Retail Invoice Matching Installation Guide* for hardware and software requirements. Also see Oracle Retail application software compatibility information.

System Assumptions

- Unit of Measure

  For invoices sent from RMS with quantities representing weight rather than number of eaches, ReIM converts the unit of measure (UOM) on the receipt to the UOM on the invoice.

- ReIM uses non-merchandise codes defined on the RMS table NON_MERCH_CODE_HEAD. The form that allows users to enter non-merchandise codes in RMS is not available when the RMS invoice match indicator (SYSTEM_OPTIONS.REIM_IND) is set to no. Instead, non-merchandise codes should be added to the NON_MERCH_CODE_HEAD table using the database.

- Supplier options

  All suppliers must have options defined for their invoices to be processed by the system, and the terms defined for those suppliers must be completely updated in RMS. To support the use of suppliers in ReIM, terms must have the following properties on the TERMS_DETAIL table:

  – ENABLED_FLAG is set to Y.

  – START_DATE_ACTIVE must be defined.

  – END_DATE_ACTIVE must be defined.

- GL account maintenance

  All reason codes, non-merchandise codes, and basic transactions must be mapped through GL account maintenance to support posting to the retailer's financial solution. Transactions are posted to a staging table in ReIM, the extract to update the accounts payable/financial solution is the retailer's responsibility.

- TAX

If TAX is turned on, the retailer must have TAX regions, TAX items, and TAX codes set up in the merchandising system (such as RMS) to support validation of invoiced TAX charges. Verify the following values on the IM_SYSTEM_OPTIONS table:

Note: The values below should not be changed after initial setup. Changing them can cause errors in the system.

- NUM_TAX_ALLOW is set to S (single) TAX, N (no) TAX.

  TAX_VALIDATION_TYPE is set to RECON (Reconcile TAX), VENDR (Always Use Vendor TAX), or RETLR (Always use Retail TAX).
- The DEFAULT_TAX_HEADER is set to Y or N.
- TAX_DOCUMENT_CREATION_LVL is set to ITEM or FULL_INVOICE.

reim.properties File

The property file has only technical configurations that are specific to the particular deployment. All the business related configurations are located within the system option scope.

The following properties are configurable

- #help settings
  - help.server.url - Help server URL
  - reim.help.library - path to the ReIM specific help deck
  - reim.release.version - help version
  - reim.help.url.suffix -help URL suffix identifying specific document within the library
  - reim.help.default.url.suffix -default page within the document
- #JMS settings
  - reim.jms.connection.factory.name -name of the JMS factory within JNDI tree on WL used for spreadsheet induction
  - reim.jms.queue.doc.induction.name - name of the JMS queue within JNDI tree on WL used for spreadsheet induction
  - reim.jms.application.code- Application code identifier used for JMS publishing
- #Notification settings
  - reim.jdbc.raf.async.task- name of the data source within JNDI tree on WL used for async publishing

Logging Configuration

Oracle Retail Invoice Matching utilizes the industry-standard Apache Log4j logging framework to log system messages and exceptions. This framework is embedded in the application code to allow for configurable logging to suit the needs of the retailer.

Please note that Log4j is used for batch clients only. Server logging is done using standard WbLogic logging infrostracture.

Log4J Conventions

The Log4j API system utilizes three main configurable entities:

- Loggers
- Appenders
- Layouts

Loggers are responsible for defining exactly what gets logged. Typically, loggers define a specific level of detail (the log level) for a specific java package name as well as an appender the logger is assigned to. These criteria are then delegated to the appropriate appender for the specific logger. A single logger can be assigned to multiple appenders.

Appenders are used to dictate where logged content is directed to for a given logger. For example, the retailer may wish to configure a log appender to publish a log to a database table, a flat file, or an e-mail address. For each of these options, a separate appender would be defined and assigned to a specific logger.

Layouts are leveraged by the appender to dictate the exact content of the log message. Relevant information may include: date, time, and origin of the error message. These values can all be configured through the log layout.

Log4J Properties

The log4j.properties file holds all of the information relevant to logging for batch clients.

Table 2–1 Log4J Parameters

Parameter	Description
log4j.appender.BATCHFILE	Type of file appender.
log4j.appender.BATCHFILE.File	Path to the log file where batches will write log entries.
log4j.appender.BATCHFILE.DatePattern	Date pattern for log file.
log4j.appender.BATCHFILE.Append	Indicator if the log should append entries to the existing file.
log4j.appender.BATCHFILE.Threshold	Logging threshold level.
log4j.appender.BATCHFILE.layout	Pattern layout for log entry.
log4j.appender.BATCHFILE.layout.ConversionPattern	Pattern for log entry.

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. ReIM has been internationalized to support multiple languages.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Reports
- Demonstration data
- Training materials

The user interface for ReIM has been translated into:

- Chinese (simplified)
- Chinese (traditional)
- Croatian
- Dutch
- French
- German
- Greek
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish
- Swedish
- Turkish

Setting the User Language

To set the language ReIM displays in, set the user language. Choose Preferences from the drop-down menu under the user name, then choose Language in the taskbar. There are two language settings you can choose:

- Default: This will permanently change the language for this user. All subsequent sessions will be in this language.
- Current Session: This will change the language only for this session. It will revert back to the Default the next time this user logs in.
- When finished making your selections, click the Save button.

Setting Date, Time, and Number Formats

ReIM allows the user to see dates and times in a format appropriate for their own locale, regardless what the data is stored in. Set the language ReIM displays in, set the user language. Choose Preferences from the drop-down menu under the user name, then choose Regional in the taskbar. There are multiple settings you can choose:

- Territory: Choose a territory from this list. ReIM will automatically pick Date, Time, and Number formats appropriate for that territory.
- Date Format: Choose an option from this menu to select a date format that is different from the default for the selected Territory.
- Time Format: Choose an option from this menu to select a time format that is different from the default for the selected Territory.
- Number Format: Choose an option from this menu to select a number format that is different from the default for the selected Territory.
- Time Zone: Choose an option from this menu to select your time zone.

When finished making your selections, click the **Save** button.

Translations

Most user interface and message translations are stored in .java files. When you select a different language from the Preferences screen, ReIM will choose the correct .java files for that language.

Some translations for some drop-down menus are stored on four database tables. The tables are RTC_LOOKUP_VALUES_TL, RTC_LOOKUP_TYPES_TL, RAF_FACET_ATTRIBUTE_CFG_TL, and RAF_NOTIFICATION_TYPE_TL. These tables are multilingual; all languages of these strings (English as well as all translations) are stored in the same place.

ReIMResources.properties

This file contains a key value pair for some of the labels visible through the GUI at run time. Text labels and error messages have been identified, separated from the core source code, and placed into the properties file. The contents of the file can be used for retailer-specific configuration purposes (such as for the creation of custom labels or error messages). Some other sources of translatable data is used within the system such as XLIF files and database tables.

batch.properties

This file contains security information about batch clients.

Table 2–2 batch.properties Parameters

Parameter	Description
providerUrl	URL of the server JNDI.
csm.wallet.partition.name	Name of the wallet partition where the batch credentials are stored.
csm.wallet.path	Path of the wallet where the batch credentials are stored

3

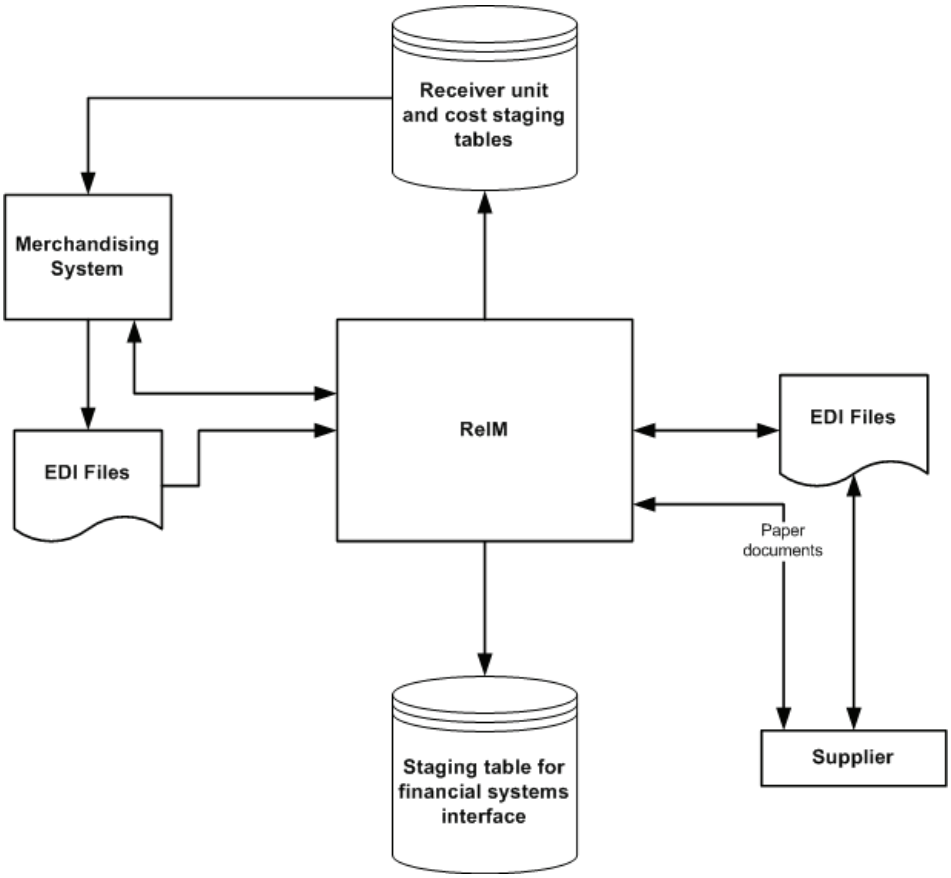
Integration

This chapter describes how ReIM integrates with other systems, related interfaces, and file layouts. It includes an integration overview, a discussion of EDI (with layouts), an explanation of how ReIM interfaces with financial systems, and a summary of LDAP user authentication.

Additional configuration may be needed to properly integrate other systems with ReIM. See Chapter 5, "System Configuration" for additional information.

Integration Overview

This section provides a diagram that shows the overall direction of the data among the applications and tables. The accompanying explanations are written from a system/staging table-to-system/staging table perspective, illustrating the movement of data.



From the Supplier (to EDI) to ReIM

ReIM receives supplier invoices and credit notes through EDI or through on-line entry processes. These document types are described later in this chapter.

From ReIM (to EDI) to the Supplier

ReIM generates debit memos, credit note requests and credit memos for various reasons. Each of these documents is recorded in ReIM tables to allow for retailer reporting. Also, a ReIM process reads these tables and creates a file of these documents to support the retailer's EDI transmissions to suppliers.

From ReIM to the Staging Table for Financial Systems Interface

For a description of the data that is sent through this interface, see "Financial System Interface" later in this chapter.

From the Merchandising System to ReIM (Directly and Through EDI)

ReIM is able to access foundation data, such as item, purchase order, supplier, and other information directly from RMS tables. ReIM provides the drivers to access these tables without further integration work.

- **Receipts**

  Receipts are records of purchased merchandise arriving at the store or warehouse. Receipt data is accessed in RMS, and certain data elements are extracted from RMS into ReIM tables to support ReIM-specific actions performed against receipts (for example, splitting receipt quantities, updating statuses, and so on).

- **Purchase Orders**

  Purchase orders (POs) are created in RMS and represent a legally binding agreement between retailer and supplier for the purchase and sale of goods. The retailer records the quantity, cost and delivery location of items from the supplier. On a single PO, RMS supports different costs for the same item going to different locations. PO costs are used to value receipt quantities.

- **Item**

  ReIM processes matches at the item transaction-level (that is, SKUs). For reference purposes, UPCs may be used, so they should be provided by the merchandising system. See Oracle Retail Merchandising System documentation for more information about the multi-level item structure.

- **Partner**

  A partner is a business that supplies and bills a retailer for non-merchandise services. Examples of partners are banks, agents, and expense suppliers. A partner cannot send merchandise invoices to retailers.

- **Valued Added Tax (TAX) Code and Rate**

  TAX is embedded in the cost of the item. ReIM provides for validation of taxes charged on the invoice against TAX codes/rates stored in RMS tables for the item.

- **Consignment**

  Consignment is an arrangement whereby the physical control of merchandise (but not the title of ownership) is transferred from one business known as the consignor (for example, the vendor) to another known as the consignee (for example, the retailer). The title to the goods remains with the consignor until the

goods are sold. When consigned goods are sold, the consignor invoices the consignee. On this invoice, the cost of each item is reduced to a certain proportion, called the consignment rate. The consignment rate, predetermined by both parties, represents the consignor's share of the sale. Once the merchandising system records a sale, a consignment invoice is created in ReIM for a percentage of the sale cost. The receipt is implied based on the consignment rate applied to the selling price; accordingly, the self-billed invoice is assumed to be in matched status.

- **Return to Vendor (RTV)**

  An RTV is a retailer-initiated purchase return of inventoried goods to an external vendor. The merchandising system uses RTV data to update inventory positions and write requisite transactions to the stock ledger. ReIM receives RTV data through the merchandising system from the store and warehouse inventory systems where it is initiated, where a charge-back document is created.

- **Deal Bill Backs**

  RMS tracks certain types of supplier deals (for example, rebates, vendor funded markdowns, and so on) for billing back to the supplier. Information to support these billings is received in ReIM through an RMS extract. ReIM creates a charge back document for these billings, which may be subject to edit/approval in ReIM or automatically processed to the financial staging table for export to the retailer's accounts payable solution, based on an RMS parameter.

- **Other Data Elements Received from RMS**
  - Non-merchandise codes
  - Currency
  - Exchange rates
  - Store/warehouse location type
  - Supplier information
  - Supplier address (invoice address, returns address, and so on)
  - Merchandise hierarchy
  - Business date
  - Terms and terms ranking

From ReIM to Receiver Unit and Cost Staging Tables to RMS

Receiver cost and unit adjustments are initiated in ReIM update receipts held in RMS tables. Receiver adjustments, resulting from the ReIM discrepancy resolution process, create cost and/quantity adjustments to receipt tables in RMS, as well as to supplier and purchase order tables for certain types of cost resolutions.

From ReIM to the Merchandising System

- **Receipt Status**

  When the entire receipt is matched (all the lines to invoices), ReIM provides and update to the invoice match status (that is, from unmatched to matched) on the shipment table in RMS.

- **Shipment (Receipts) Table Quantity Matched Update**

When ReIM matches a portion and/or all of a receipt line to an invoice line, ReIM makes a corresponding update to the quantity matched column.

Electronic Data Interchange (EDI) Tables and Files

Electronic Data Interchange (EDI) facilitates the computer-to-computer transmission of business information and transactions, such as invoices and purchase orders. EDI represents a convenient method by which a retailer and its suppliers can transfer information back and forth. The Voluntary Interindustry Commerce Standard (VICS) EDI is used by the general merchandise retail industry.

ReIM has two file-based EDI interfaces. Note that neither follows the VICS EDI standard. The ReIM EDI interfaces have been customized, and the retailer must translate them.

The interfaces represent the upload of invoices or other documents from a supplier or another application and the download of documents to suppliers. These two common types of EDI are described below:

- EDI Injector is the standard description for an EDI process that uploads documents.
- EDI invoice download is the standard description for an EDI process that downloads Debit Memo, Credit Note Request, and Credit Memo data from ReIM to suppliers.

For information about ReIM batch processes related to both of these types of EDI, see "Chapter 6, "Batch Processes"". Note that although the majority of invoices are created through either EDI Injector or batch entry, users can also create invoices online and add details, or use the online dialog to add details to an invoice that was EDI uploaded.

The EDI Reject Table

The EDI Injector (reimediinjector) batch process uploads invoices and credit notes from EDI files into the Injector workspace tables (IM_INJECT_xxx), validates the data against a set of rules, and then moves valid documents into the operational tables. This process validates the information in the file against itself and against RMS/ReIM database. A limited set of data validation errors can cause the invalid transaction to remain in the Injector workspace tables (IM_INJECT_DOC_xxx) in fixable status where the data can be corrected through an online process.

The following errors can be manually corrected through the front end:

- Supplier number (or Partner ID): This value must be a valid supplier site (SUPS table) or partner (PARTNER table) in RMS (or the equivalent merchandising system).
- Order numbers: Order numbers must be approved and created for the supplier or linked suppliers in RMS (or the equivalent merchandising system) on the ORDHEAD table. Non-merchandise invoices may not have any order numbers associated, so this validation should be skipped for this type of invoice.
- Order/location combination: The system validates that all order number/location combinations in the file are valid within RMS or the equivalent merchandising system (meaning that the relationship must exist on the ORDLOC table).
- Terms code: All terms must exist within RMS or the equivalent merchandising system on the TERMS table.

- Invoice date: A document cannot be older than the VDATE minus the post-dated document days' system level parameter value or newer than the VDATE.
- Merchandise invoices cannot be associated with a partner; they must only be associated with a supplier.
- Credit notes from a partner cannot have item records attached unless the partner type is a manufacturer, distributor, or wholesaler (type S1, S2, or S3).

The EDI Reject File

A limited set of data validation errors (identified in the file layout Validation column) cause the invalid transaction to be written to the reject file (named by the retailer).

EDI Injector File Layout (Based on EDI 810)

The following describes the input and output specification for the EDI Injector File.

All Files Layouts Input and Output

Input file format:

FHEAD (1): Start of file.

THEAD (1…n): Transaction (document) level info. Each file must have at least 1 THEAD.

TDETL (0…n): Item detail records for this transaction. TDETL is required for Merchandise Invoices and optional for Non-Merchandise Invoices, Credit Note documents, and debit memo documents.

TALLW (0…n): Allowance records for this item. TALLW is optional.

TNMRC (0…n): Non-merchandise records for this transaction. Required on non-merchandise documents.

TVATS (0…n): VAT breakdown by VAT code. TVATS is optional.

TTAIL (1…n): Marks the end of a THEAD record. Each THEAD requires exactly one TTAIL.

FTAIL (1): Marks the end of the file.

TDETL and TNMRC do not need to occur in order. TALLW must follow TDETL

If records are encountered in any order other than specified above, execution of program will halt.

Example:

FHEAD

THEAD

TNMRC

FTAIL (no TTAIL encountered)

If a record descriptor is encountered other than those specified in this document, execution of program will halt.

Reject file will have an identical format. If no records are rejected, it will consist of only the FHEAD and FTAIL lines.

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Null variables should be blank.

Single location invoices will be inserted into IM_DOC_HEAD, IM_INVOICE_DETAIL and IM_DOC_NON_MERCH.

It is assumed all values that have dependent information included in the file (for example, location has dependent information of order no, upc, upc-supp, and so on) are valid for the RMS system. The following is never anticipated to happen: only locations A, B, and C exist in RMS; EDI reads a transaction that has location D. This sort of file may not be flagged as invalid in any way.

Uploaded documents with details must have at most one associated UPC, item or VPN identifier. When system Tax processing is enabled, documents that fail to meet these criteria will be rejected to the file by the EDI Injector Batch process. When Tax is disabled, the document will be available for review and correction through the Invoice Matching user interface in the EDI Maintenance screen.

FHEAD - File Header. First record of an upload file.

Field Name	Field Type	Description	Req	Validation
Record Descriptor	Char(5)	Describes file record type	Y	Halt execution if not FHEAD.
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not 0000000001.
Gentran ID	Char(5)	The type of transaction this file represents.	Y	Halt execution if not UPINV.
Current date	Char(14)	File date in YYYYMMDDHH24MISS format.	Y	Halt execution if invalid date format.

THEAD - Transaction Header. Start of a document transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type	Y	THEAD
Line id	Number(10)	Sequential file line number	Y	Halt execution if not in sequence
Transaction number	Number(10)	Sequential transaction number. All records within this transaction will also have this transaction number.	Y	Reject entire file if: transaction number is not numeric or not in sequence first transaction number is not 0000000001

Field Name	Field Type	Description	Req	Validation
Document Type	Char(6)	Describes the type of document being uploaded. The document type will determine the types of detail information that are valid for the document upload. Stored in IM_DOC_HEAD.TYPE. Valid values are: MRCHI - Merchandise Invoice NMRCHI - Non Merchandise Invoice CRDNT - Credit Note DBMC - Debit Memo Cost DBMQ - Debit Memo Qty CRDMC - Credit Memo Cost CNRC- Credit Note Request Cost CNRQ- Credit Note Request Qty	Y	Reject transaction to file if document type is null document type is not MRCHI (merchandise invoice), NMRCHI (non-merchandise invoice), CRDNT (credit note), DBMC (Debit Memo-Cost), DBMQ (Debit Memo-Qty), CNRC (Credit Note Request-Cost), CNRQ (Credit Note Request-Qty), CRDMC (Credit Memo Cost) document type is CRDNT (credit note); vendor is not a supplier, manufacturer, distributor, or wholesaler. document type is CRDNT and TALLW records exist document type is MRCHI and item detail records DO exist for this transaction (this type of transaction must have no item detail records) document type is CRDNT,NMRCHI, DBMC, DBMQ, CRDMC, CNRC, CNRQ and any error occurs with the document
Vendor Document Number	Char(30)	Vendor's document number. Stored in IM_DOC_HEAD.EXT_DOC_ID with all characters converted to their upper case (for example, ThisDocId -> THISDOCID).	Y	Reject entire upload file if the same vendor document number occurs more than once in the file. Reject transaction to file if: ■ Vendor document number is null. ■ Vendor document number is not unique for this vendor. ■ Vendor document number is not alphanumeric and the property INVOICE_NUMBER_VALIDATION_REGULAR_EXPRESSION in reim.properties is commented out. ■ Vendor document number contains special characters that are not specified in the property INVOICE_NUMBER_VALIDATION_REGULAR_EXPRESSION in reim.properties and the property is not commented out. ■ Vendor document number has leading zero and the property INVOICE_NUMBER_VALIDATION_ALLOW_ZERO in reim.properties is commented out or set to false.

Field Name	Field Type	Description	Req	Validation
Group ID	Char(10)	The Group ID is an informational field, which can be used to identify groups of invoices that were transmitted to ReIM together.	N	Reject transaction to file if: ■ Group ID exists and is non-numeric. ■ Group ID exists and is numeric and negative.
Vendor Type	Char(6)	Type of vendor (either supplier or partner) for this document. Stored in IM_DOC_ HEAD.VENDOR_TYPE Valid values are: SUPP - Supplier BK - Bank AG - Agent FF - Freight Forwarder IM - Importer BR - Broker FA - Factory AP - Applicant CO - Consolidator CN - Consignee S1 - Merch Supp level 1 S2 - Merch Supp level 2 S3 - Merch Supp level 3	Y	Reject transaction to file if: ■ Vendor type is null or if it is not a valid vendor type (from Vendor class). ■ Document type is MRCHI (merchandise invoice) and vendor type is not Supplier (SUPP).
Vendor ID	Char(10)	Vendor for this document (if a supplier, this field must be a supplier site) . Stored in IM_DOC_ HEAD.VENDOR	Y	Reject transaction to tables if: ■ Vendor is a supplier and supplier site is not valid. ■ Vendor is a supplier and vendor ID is not completely numeric.
Vendor Document Date	Char(14)	Date document was issued by the vendor (in YYYYMMDDHH24MISS format). Stored in IM_ DOC_HEAD.DOC_DATE	Y	Reject transaction to file if: ■ Vendor document date is null. ■ Date is not a valid date format. Reject transaction to tables if Vendor Document Date is: ■ After the vdate. ■ Before (vdate - post_dated_doc_ days) (from im_system_options).

Field Name	Field Type	Description	Req	Validation
Order Number/ RTV Order Number	Number(12)	Merchandising system order number for this document. Required for merchandise invoices and optional for others. Store in IM_DOC_HEAD.ORDER_NO. This field can also contain the RTV order number if the RTV flag is 'Y'	N	Reject transaction to file if: ■ Order/RTV order number exists and is not numeric. ■ Order/RTV order number is null and vendor type is a supplier. ■ Order/RTV order number is null and deal_id is null. ■ Order/RTV order number exists and vendor type is NOT a supplier. ■ Order/RTV order number exists and location or location type are null. Reject transaction to tables if RTV flag is null or 'N' AND: ■ Order number exists but is not valid for the supplier or the supplier's linked suppliers. ■ Order number exists but is not valid for the location/location type. Reject transaction to file if RTV flag is 'Y' AND: ■ RTV order number exists but is not valid for the supplier or the supplier's linked suppliers. ■ RTV order number exists but is not valid for the location/location type.
Location	Number(10)	Merchandising system location for this document. Required for merchandise invoices and optional for others. Stored in IM_DOC_HEAD.LOCATION.	Y	Reject transaction to file if: ■ Location or location type do not exist. ■ Location exists and is not numeric. ■ Location exists and location type is not Store or Warehouse. Reject transaction to tables if Location and Location Type exist but are not valid.
Location Type	Char(1)	Merchandising system location type (either Store or Warehouse) for this document. Required for merchandise invoices and optional for others. Stored in IM_DOC_HEAD.LOC_TYPE.	N	Reject transaction to file if Location type exists and location is null.

Field Name	Field Type	Description	Req	Validation
Terms	Char(15)	Terms of this document. If terms are not provided, the vendor's default terms are associated with this record. Stored in IM_DOC_HEAD.TERMS. This value is used to get the Terms Discount Percentage to be stored on IM_DOC_HEAD.TERMS_DSCNT_PCT.	N	Reject transaction to tables if Terms exist and are not valid.
Due Date	Char(14)	Date the amount due is due to the vendor (YYYYMMDDHH24MISS format). If due date is not provided, default due date is calculated based on vendor and terms. Stored in IM_DOC_HEAD.DUE_DATE.	N	Reject transaction to file if: ■ Due date exists and is not a valid date format. ■ Due date is before the vendor document date.
Payment method	Char(6)	Method for paying this document. Stored in IM_DOC_HEAD.PAYMENT_METHOD.	N	Reject transaction to file if Payment method exists and is not valid.
Currency code	Char(3)	Currency code for all monetary amounts on this document. Stored in IM_DOC_HEAD.CURRENCY_CODE.	Y	Reject transaction to file if: ■ Currency code is null. ■ Currency code is not valid. ■ Order number exists and currency code does not match the order's currency.
Exchange rate	Number (12,4)	Exchange rate for conversion of document currency to the primary currency. Stored in IM_DOC_HEAD.EXCHANGE_RATE.	N	Reject transaction to file if Exchange rate exists and is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total cost amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Cost	Number(20,4)	Total document cost, including all items and costs on this document. This value is in the document currency. Stored in IM_DOC_HEAD.TOTAL_COST and IM_DOC_HEAD.RESOLUTION_ADJUSTED_TOTAL_COST.	N	Reject transaction to file if: ■ Total cost is null. ■ Total cost is not numeric. ■ Total cost does not equal the sum of extended costs for all item detail records in this transaction. ■ Total cost is not negative and vendor document type is CRDNT.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
Total VAT Amount	Number(20,4)	Total VAT amount, including all items and costs on this document. This value is in the document currency.	N	Treat as zero if null. Reject transaction to file if: ■ Total Tax amount is not null but is not numeric. ■ Total Tax amount does not equal the sum of Tax for all item detail records PLUS the sum of Tax for all non-merch items in this transaction PLUS the sum of Tax for all allowances in this transaction.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Quantity	Number(12,4)	Total quantity of items on this document. This value is in EACHES (no other units of measure are supported in ReIM). Stored in IM_DOC_ HEAD.TOTAL_QTY and IM_DOC_ HEAD.RESOLUTION_ ADJUSTED_TOTAL_QTY.	Y	Reject transaction to file if: ■ Total quantity is null. ■ Total quantity is not numeric. ■ Total quantity does not equal the sum of quantities for all item detail records in this transaction. ■ Total quantity is not zero when vendor document type is 'NMRCHI'.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total Tax amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Total Discount	Number(12,4)	Total discount applied to this document. This value is in the document currency. Stored in IM_ DOC_HEAD.TOTAL_ DISCOUNT	Y	Reject transaction to file if: ■ Total discount is null. ■ Total discount is not numeric.
Freight Type	Char(6)	The freight method for this document.	N	Reject transaction to file if Freight type exists and is not valid.
Paid Ind	Char(1)	Indicates if this document has been paid. Stored in IM_DOC_ HEAD.MANUALLY_ PAID_IND.	Y	Reject transaction to file if: ■ Paid ind is null. ■ Paid ind is not Y or N.
Consignment indicator	Char(1)	Y	Y	Reject transaction to file if: ■ Consignment indicator is null ■ Consignment indicator is not Y or N Do not reject transaction to table if Consignment is Y.
Deal Id	Number(10)	Deal Id from RMS if this invoice is a deal bill back invoice.	N	If Deal Id is not null, Deal Approval indicator must be 'M' or 'A'. Do not reject transaction to table if deal id is not null.
Deal Detail Id	Number(10)	Deal Detail Id from RMS and is stored in IM_DOC_ HEAD. DEAL_DETAIL_ID	N	

Field Name	Field Type	Description	Req	Validation
Deal Approval Indicator	Char(1)	Indicates if the document on IM_DOC_HEAD is to be created in Approved or Submitted status.	N	Reject to file if not blank, 'M' Submitted status or 'A' approved. Do not reject transaction to table if value is not null.
RTV indicator	Char(1)	Indicates if this invoice is a RTV invoice.	Y	Reject transaction to file if: ■ RTV indicator is null ■ RTV indicator is not Y or N Do not reject transaction to table if RTV is Y.
Custom Document Reference 1	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_1.	N	
Custom Document Reference 2	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_2.	N	
Custom Document Reference 3	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_3.	N	
Custom Document Reference 4	Char(90)	This optional field is included in the upload file for client customization. No validation is performed on this field. Stored in IM_DOC_HEAD.CUSTOM_REF_4.	N	
Cross-reference document number	Number(10)	Document that a credit note is for. Blank for all document types other than merchandise invoices. Stored in IM_DOC_HEAD.REF_DOC.	N	Reject transaction to file if Cross-reference document number exists and is not numeric
Reference Invoice Number	Char(50)	This optional field is included in the upload file for the reference of original invoice. This value is stored in IM_DOC_HEAD.REF_INV_EXT_DOC_ID	N	
Reference Credit Note Request Number	Char(50)	This optional field is included in the upload file for the reference of original credit note request. This value is stored in IM_DOC_HEAD. REF_CNR_EXT_DOC_ID	N	

TVATS - VAT breakdown by VAT code. This information is inserted in IM_DOC_TAX

Field Name	Field Type	Description	Req	Validation
Field record descriptor	Char(5)	Marks costs at Tax rate line.	Y	TVATS Reject entire transaction to file if this type of record exists and the transaction has any error. See technical design for additional validations. Reject to file if in im_system_options Tax is on, but there is no TVATS.
Line id	Char(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)		Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
VAT code	Char(6)	VAT code that applies to cost.	Y	Reject to file if VAT code is not valid.
VAT rate	Number(20,10)	VAT Rate corresponding to the Tax code.	Y	Reject to file if VAT rate is not numeric.
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Cost at this VAT code	Number(20,4)	Total amount that must be taxed at the above VAT code.	Y	Reject to file if not numeric.

TDETL - Item Detail Record. This information is inserted into the IM_INVOICE_DETAIL table for Merchandise Invoice and IM_DOC_DETAIL_REASON_CODES for Credit Notes.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TDETL
Line Id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for this item detail record.	Y	Reject to file if Tax rate is not numeric ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.

Field Name	Field Type	Description	Req	Validation
UPC	Char(25)	UPC for this detail record. Valid item number is retrieved for the UPC. Stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM.	Y Exclusive with item	Reject transaction to file if: ■ UPC is null and Item is null. ■ Both UPC and Item are not null. Reject transaction to tables if: ■ Valid item is not found for UPC and UPC supp. ■ Valid item is not associated with the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items).
UPC Supplement	Number(5)	Supplement for the UPC. Note: UPC Supp is only valid for 9.0 implementation. For 10.1 implementation, this field will ALWAYS be blank.	N	Reject transaction to file if: ■ UPC supplement exists and UPC doesn't exist. ■ UPC supplement exists and is not numeric.
Item	Char(25)	Item for this detail record. Item number is verified and stored in IM_INVOICE_DETAIL.ITEM or IM_DOC_DETAIL_REASON_CODES.ITEM.	Y Exclusive with UPC	Reject transaction to file if: ■ UPC is null and Item is null. ■ Both UPC and Item are not null. ■ Valid item is not associated with the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items).
VPN	Char(30)	Vendor Product Number provided by the supplier. It is used to identify an item when an item number has not been provided. VPN is displayed on the Invoice Maintenance screen and may be used during the on-line matching process.	Y (exclusive with item and UPC)	Reject transaction to file if: ■ VPN is null and UPC is null and Item is null. ■ At least two of the following are not null: UPC, VPN and ITEM. Reject transaction to tables if: ■ Valid item is not found for VPN for the supplier. ■ The item found is identical to another detail item for this transaction (no duplicate items). ■ There are multiple items for the supplier with the VPN provided and: no items on the PO for the document OR multiple items on the PO for the document.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
OriginalDocument Quantity	Number(12,4)	Quantity, in EACHES, of the item on this detail record. Stored in IM_INVOICE_DETAIL.INVOICE_QTY and IM_INVOICE_DETAIL.RESOLUTION_ADJUSTED_QTY.	Y	Reject transaction to file if: ■ Original document quantity is null. ■ Original document quantity is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Unit Cost amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Original Unit Cost	Number(20,4)	Unit cost, in document currency, of the item on this detail record. Stored in IM_INVOICE_DETAIL.UNIT_COST and IM_INVOICE_DETAIL.RESOLUTION_ADJUSTED_UNIT_COST.	Y	Reject transaction to file if: ■ Original unit cost is null. ■ Original unit cost is not numeric.
Original VAT Code	Char(6)	Tax code for item.	Y	Reject to file if VAT code is invalid.
Original VAT rate	Number(20,10)	Tax Rate for the Tax code/item.	Y	Reject to file if VAT rate is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	Reject transaction to file if: ■ Sign indicator is null. ■ Sign indicator is not + or -.
Total Allowance	Number(20,4)	Sum of allowance details for this item detail record. If no allowances exist for this item detail record, value is 0.	Y	Reject transaction to file if: ■ Total allowance is null. ■ Total allowance is not numeric. ■ Total allowance does not equal the sum of allowance amounts for all allowance records in this item detail record. ■ Total allowance is not 0 and vendor document type is CRDNT.

TALLW - Allowance Record. This information is inserted into IM_INVOICE_DETAIL_ALLOWANCE table.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TALLW
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.

Field Name	Field Type	Description	Req	Validation
Transaction Number	Number(10)	Transaction number for this item allowance record.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Allowance Code	Char(6)	Allowance code for this allowance record. Stored in IM_INVOICE_ DETAIL_ ALLOWANCE.ALLOWA NCE_CODE.	Y	Reject transaction to file if: ■ Allowance code is null. ■ Allowance code is not valid.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) allowance amount.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.
Allowance Amount	Number (20,4)	Amount of allowance in document currency. Stored in IM_INVOICE_ DETAIL_ ALLOWANCE.ALLOWA NCE_AMOUNT.	Y	Reject transaction to file if allowance amount is null or not numeric.
Allowance VAT Code	Char(6)	VAT Code for Allowance.	Y	Reject to file if VAT code is not valid.
Allowance VAT rate at this VAT code	Number (20,10)	VAT Rate corresponding to theVAT code.	Y	Reject to file if not numeric.

TNMRC - Non-Merchandise Record. Records of this type will contain non-merchandise costs. These costs are inserted into the IM_DOC_NON_MERCH table. Non-merchandise costs records are only required when the document type is non-merchandise. Non-merchandise cost records are also associated with merchandise type documents if the vendor associated with the document allows non-merch costs on merchandise invoices (IM_SUPPLIER_OPTIONS. MIX_ MERCH_NON_MERCH_IND).

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TNMRC
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for this non-merchandise record.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Non Merchandise Code	Char(6)	Non-Merchandise code that describes this cost. Stored in IM_DOC_ NON_MERCH.NON_ MERCH_CODE.	Y	Reject transaction to file if: ■ Non-merchandise code is null. ■ Non-merchandise code is not valid.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) Non Merchandise Amt.	Y	Reject transaction to file if sign indicator is null or if it is not + or -.

Field Name	Field Type	Description	Req	Validation
Non Merchandise Amt	Number(20,4)	Cost in the document currency. Stored in IM_DOC_NON_MERCH.NON_MERCH_AMT.	Y	Reject transaction to file if: ■ Non-merchandise amount is null. ■ Non-merchandise amount is not numeric. ■ Non-merchandise amount does not have a negative value and this is part of a credit note document (THEAD.Vendor Document Type = CRDNT).
Non Merch VAT Code	Char(6)	VAT Code for Non-Merchandise.	Y	Reject to file if VAT code is not valid.
Non Merch VAT code at this VAT code	Number(20,10)	VAT Rate corresponding to the VAT code.	Y	Reject to file if not numeric.
Service Performed Indicator	Char(1)	Indicates if a service has actually been performed. Stored in IM_DOC_NON_MERCH.SERVICE_PERF_IND.	Y	Reject transaction to file if: ■ Service performed indicator is null. ■ Service performed indicator is not Y or N.
Store	Number(10)	Store at which the service was performed. Stored in IM_DOC_NON_MERCH.STORE.	N	Reject transaction to file if: ■ Store exists and is not numeric. ■ Service performed indicator is Y and store is not valid.

TTAIL - Transaction Tail. Marks the end of a transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	TTAIL
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Transaction number	Number(10)	Transaction number for the transaction that this record is closing.	Y	Reject entire file if: ■ Transaction number is not numeric. ■ Transaction number is not the same as the current transaction.
Transaction lines	Number(6)	Total number of detail lines within this transaction.	Y	Reject transaction to file if transaction lines is not numeric, if it does not match the count of lines within the transaction, or if it is zero (transaction must have details).

FTAIL - File TAIL. Marks the end of the upload file.

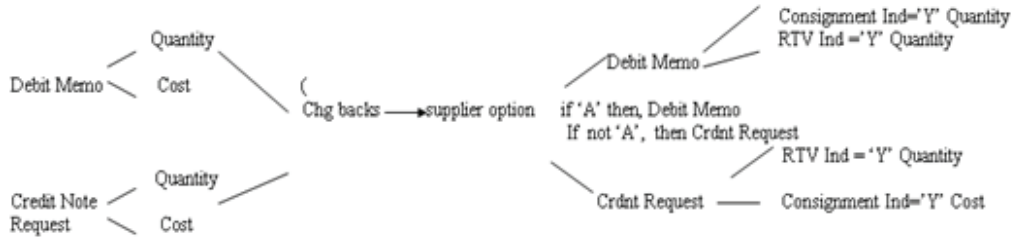
Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	FTAIL

Field Name	Field Type	Description	Req	Validation
Line id	Number(10)	Sequential file line number.	Y	Halt execution if not in sequence.
Number of lines	Number(10)	Total number of lines within this file not counting FHEAD and FTAIL.	Y	Halt execution if number of lines is not numeric, if it does not match the count of lines within the file (excluding FHEAD and FTAIL), or if it is 2 (FHEAD and FTAIL only, file has no transactions).

Notes

Consider the following.

- The EDI Injector Batch process has the ability to recognize only a new document type. In FHEAD of the EDI flat file, the Document Type does not include CRDMC (credit memo cost). When the document type in the flat file is Debit Memo Cost, Debit Memo Qty, Credit Note Request Cost, or Credit Note Request Qty, and if the amount (Total Cost) for a Deal Charge Back Document that is sent over from RMS is negative a Credit Memo Cost is created.
- For charge back documents, use the following flow chart to determine what document type to be populated in the database.



- If the document type is merchandise invoice and the consignment indicator is Y, the status is matched; if the consignment indicator is not Y, the status isready for match; if the document type is not merchandise invoice, the status is approved.
- If the consignment indicator is Y, then set the terms to Due Immediately terms (term ID = 48), and set the terms discount percentage to 0.
- That Tax codes and rates in the detail of documents are those known for the item and location when the document is not an import Document. Given a combination of TDETL.item and location, we could find a Tax. The Tax code and Tax rate in the Tax should be the same as the original Tax code and original Tax rate in the TDETL.
- The merchandises header Tax and detail Tax are consistent (for example, Tax basis by Tax rate and Tax amount by Tax rate). Total header Merchandise Tax information is calculated from total document Tax information and Tax information for non merchandise costs. For example, for each Tax Code in TDETL and TNMRC: Thead.Total VAT Amount at this Tax code = total Tax from TDETL at this Tax code + total Tax from TNMRC at this Tax code. Total Tax from TDETL at this Tax code = sum(original document quantity * original unit cost * original Tax rate). Total Tax from TNMRC at this Tax code =sum(Non Merch Tax rate * Non Merch Amt).Thead.Total VAT Amount at this VAT code = sum(TVATS.Vat rate * TVATS.cost at this VAT code).

- For an EDI upload document, if the Tax Region of the header is different from the Tax region of the supplier, it is an import document. Import document will not contain Tax information. (LocVatRegion != SupplierVatRegion, then it is an import document). If a document is not an import document, plus the system_option.vat is on; if the TVATS is null, reject to file.
- To decide whether a Tax code is valid in the TDETL, first find the Tax code given the information of item and location. If they are equal, then the Tax code is valid; if they are not equal, check if the Tax code exists in the effective Tax codes; if the Tax code exists, then it is valid but is populated to the audit table.
- If RTV indicator or consignment indicator is Yes and deal ID is not null, it must reject to file.
- If Item field is populated and there is an error it should always reject to file. In order to reject to the tables, we must have the UPC field populated and not the Item field.

EDI Invoice Download File Layout (Based on EDI 812)

Output file format:

FHEAD (1): Start of file.

THEAD (1…n): Transaction (document) level info. Each file must have at least 1 THEAD.

TDETL (0…n): Item detail records for this transaction.

TNMRC (0…n): Non-merchandise records for this transaction.

Required on non-merchandise documents, optional otherwise.

TVATS (0…n): Doc Tax detail records for this transaction, optional.

TTAIL (1…n): Marks the end of a THEAD record. Each THEAD requires exactly one TTAIL.

FTAIL (1): Marks the end of the file.

If records are encountered in any order other than specified above, execution of program will halt.

Example:

FHEAD

THEAD

TNMRC

TVATS

FTAIL (no TTAIL encountered)

If a record descriptor is encountered other than those specified in this document, execution of program will halt.

All character variables should be right-padded with blanks and left justified; all numerical variables should be left-padded with zeroes and right-justified. Null variables should be blank.

Note: The file is not threaded, but rather ordered by vendor id (THEAD). It is assumed that this file is broken out by vendor id during the translation process.

FHEAD - File Header. First record of an upload file.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	FHEAD	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Gentran ID	Char(5)	DNINV	Y	
Current date	Char(14)	File date in YYYYMMDDHH24MISS format.	Y	

THEAD - Transaction Header. Start of a document transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	THEAD	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Sequential transaction number. All records within this transaction will also have this transaction number.	Y	
Document Type	Char(6)	Describes the type of document being downloaded. The document type will determine the types of detail information that are valid for the document downloaded. Retrieved from IM_DOC_HEAD.TYPE where type is debit memo, credit note request or credit memo and in Approved or Posted Status.	Y	
Vendor Document Number	Char(30)	Vendor's document number. Retrieved from IM_DOC_HEAD.EXT_DOC_ID.	Y	
Invoice Number	Char(6)	Corresponding invoice resolved by the document. Retrieved from IM_DOC_HEAD.REF_DOC.	Y	
Vendor ID	Number(10)	Vendor for this document. Retrieved from IM_DOC_HEAD.VENDOR	Y	
Document Date	Char(14)	Date the document was entered into the system in YYYYMMDDHH24MISS format. Retrieved from IM_DOC_HEAD.DOC_DATE	Y	
Order	Number(10)	Order number for this document, if any. Retrieved from IM_DOC_HEAD.ORDER_NO	N	
Location	Number(10)	Location for this document, if any. Retrieved from IM_DOC_HEAD.LOCATION.	N	

Field Name	Field Type	Description	Req	Validation
Location Type	Char(1)	Location type for this document, if any. Retrieved from IM_DOC_HEAD.LOC_TYPE.	N	
Terms	Char(15)	Terms of this document. Retrieved from IM_DOC_HEAD.TERMS.	N	
Due Date	Char(14)	Date the amount due is due from the vendor (YYYYMMDDHH24MISS format). Retrieved from IM_DOC_HEAD.DUE_DATE.	N	
Currency Code	Char(3)	Currency code for this document. Retrieved from IM_DOC_HEAD.CURRENCY_CODE.	N	
Exchange Rate	Number(12,4)	Exchange rate for conversion of document currency to the primary currency. Retrieved from IM_DOC_HEAD.EXCHANGE_RATE.	N	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) total cost.	Y	
Total Cost	Number(20,4)	Total document cost, including all items and costs on this document. This value is in the document currency. Retrieved from IM_DOC_HEAD.TOTAL_COST.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) total VAT amount	Y	
Total VAT Amount	Number(20,4)	Total VAT amount, including all items and costs on this document. This value is in the document currency.	N	
Sign indicator	Char(1)	Indicates a positive (+) or negative (-) quantity	Y	
Total Quantity	Number(12,4)	Total quantity of items on this document. This value is in EACHES (no other units of measure are supported in ReIM). Retrieved from IM_DOC_HEAD.TOTAL_QTY.	Y	

TDETL - Item Detail Record. This information is inserted into the IM_DOC_DETAIL_REASON_CODES table.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TDETL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Generated Transaction number for this item detail record	Y	
Item	Char(25)	Internal SKU/Item for this document. This is always sent. Retrieved from IM_DOC_DETAIL.ITEM.	Y	

Field Name	Field Type	Description	Req	Validation
UPC	Char(25)	UPC for this detail record. Retrieved from UPC_EAN.UPC (RMS 9.0) or ITEM_MASTER.ITEM (RMS 10.1). This field is sent if available. Note: UPC is used for RMS 9.0 and Ref-Item is used for RMS 10.1. Ref-Item consists of UPC and UPC-Supp appended together with a separating hyphen(-).	N	
UPC Supplement	Number(5)	Supplement for the UPC. Retrieved from UPC_EAN.UPC_SUPPLEMENT. This field is sent if available. Note: UPC Supp is only valid for 9.0 implementation. For 10.1 implementation, this field will always be blank.	N	
VPN	Char(30)	Vendor Product Number. This field is sent if available. Retrieved from ITEM_SUPPLIER.VPN.	N	
Comments	Char(200)	Comments associated with Reason Code. Retrieved from IM_DOC_DETAIL_COMMENTS.TEXT	Y	
Reason Code	Char(6)	Reason Code for this document. Retrieved from IM_DOC_DETAIL_REASON_CODES.REASON_CODE_ID	Y	
Reason Code description	Char(50)	Description associated with Reason Code. Retrieved from IM_REASON_CODES.REASON_CODE_DESC		
Sign indicator	Char(1)	Indicates a positive (+) discrepant qty.	Y	
Discrepant Quantity	Number(12,4)	Quantity, in EACHES, of the item that is discrepant for this detail record. Retrieved from IM_DOC_DETAIL_REASON_CODES.ADJUSTED_QTY.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) discrepant cost.	Y	
Discrepant cost	Number(20,4)	Unit cost, in document currency, of the item that is discrepant for this detail record. Retrieved from IM_DOC_DETAIL_REASON_CODES.ADJUSTED_UNIT_COST.	Y	
Original VAT code	Char(6)	VAT code for item.		
Original VAT rate	Number(20,10)	VATRate for the VAT code/item.		

TNMRC - Non-Merchandise Record. Records of this type will contain non-merchandise costs. These costs are retrieved from the IM_DOC_NON_MERCH table. Non-merchandise cost records are only required when the document type is non-merchandise. Non-merchandise cost records are also associated with merchandise type documents if the vendor associated with the document allows non-merch costs on merchandise invoices (IM_SUPPLIER_OPTIONS. MIX_MERCH_NON_MERCH_IND).

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TNMRC	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Transaction number	Number(10)	Generated Transaction number for this non-merchandise record.	Y	
Non Merchandise Code	Char(6)	Non-Merchandise code that describes this cost. Retrieved from IM_DOC_ NON_MERCH.NON_MERCH_CODE.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) non merchandise amount.	Y	
Non Merchandise Amt	Number(20,4)	Cost in the document currency. Retrieved from IM_DOC_NON_MERCH.NON_ MERCH_AMT.	Y	
Non Merch VAT code	Char(6)	VAT Code for Non_merchandise.	Y	
Non Merch VAT code at this VAT code	Number(20,10)	VATRate corresponding to the VAT code.		

TVATS - VAT Detail record.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TVATS	Y	
Line id	Number(10)	Sequential line number.	Y	
Transaction number	Number(10)		Y	
VAT code	Char(6)	VAT code that applies to cost.	Y	
VAT rate	Number(20,10)	VAT Rate corresponding to the VAT code.	Y	
Sign indicator	Char(1)	Indicates either a positive (+) or a negative (-) Original Document Quantity amount.	Y	
VAT Basis	Number(20,4)	Total amount that must be taxed at the above VAT code.	Y	

TTAIL - Transaction Tail. Marks the end of a transaction.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TTAIL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Line number	Number(10)	Generated Transaction number for the transaction that this record is closing.	Y	
Transaction lines	Number(6)	Total number of detail lines within this transaction.	Y	

FTAIL - File TAIL. Marks the end of the upload file.

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	FTAIL	Y	
Line id	Number(10)	Generated Sequential file line number.	Y	
Number of lines	Number(10)	Total number of lines within this file, not including FHEAD and FTAIL.	Y	

Document Induction via UI

The application allows injecting spreadsheets in ODS format with document data to be processed similarly to EDI injector flow. ODS files are spreadsheets in ODF data format. This format is a binary format and is nothing but archive (zip) of actual data in XML format.

Out of the box the application will have a set of templates defined within the system. The templates will be grouped by a category. At the same time retailers will be able to define custom templates.

Template categories will be document type based. There will be one or more Merchandising Invoice templates, one or more Credit Note templates, and so on.

The purpose of templates is to define the set of worksheets that are suitable for document entry process. Each worksheet will have a set of fields. While some worksheets are mandatory for a particular document type, some may be optional and as such can be removed by customization process to better suite retailer business process. Same logic applies to worksheet fields. While some of the fields are mandatory, some are optional and can be removed by customization.

Customization will be done via backend. Retailers will be able to delete base installed template. It will be up to retailers to guarantee that all the required templates are present and correct, as well as that they will not conflict with existing templates.

Template definition will be done using existing data model.

The template type will match existing document types that are allowed to be manually entered.

- NMRCHI
- MRCHI
- DEBMEQ
- DEBMEC
- CRDNRQ
- CRDNRC
- CRDMEQ
- CRDMEC

The main difference for default template definitions will be presence or lack thereof of tax related information. Another difference will be the presence or lack thereof of detail records.

Templates are defined in the following tables

- S9T_TEMPLATE - template table

- S9T_TMPL_WKSHT_DEF - template worksheets table
- S9T_TMPL_COLS_DEF- template worksheet columns table.

The template data structure is shared by multiple applications. ReIM templates are defined under DOCS9T category.

The order of the columns within the worksheet is NOT customizable and is predefined within the application. The order of the worksheets, on the other hand can be changed.

Processing of the spreadsheets is done using the same set of rules as EDI file processing. Errors identified during processing will be included in an error worksheet within the rejection file that can be retrieved via UI.

Financial System Interface

ReIM exports data to financial staging tables. This section describes these tables.

Foundation Financial Data Overview

The following types of financial information are imported in ReIM:

- Terms ranking data
- Variable department/class account segments
- Variable company/location account segments

Terms ranking information is used in the best terms calculation to choose the best term for each document. This best terms information is posted to the financial system.

Variable department/class and company/location segments are used to determine the account segments to which a document is posted.

The retailer is responsible for populating variable department/class and company/location segments. No API is provided.

Location Account Segments

ReIM uses location account segments in general ledger (GL) account mappings. The location account segments are accessed and maintained through the ReIM user interface. The segments are dynamically assigned during posting, based on the location on the invoice. The data is stored in the location account segments table (IM_DYNAMIC_SEGMENT_LOC).

Department/Class Account Segments

ReIM uses department/class account segments in GL account mappings. The department/class account segments are accessed and maintained through the ReIM user interface. The segments are dynamically assigned during posting, based on the item on the invoice. The data is stored in the department/class account segment table (IM_DYNAMIC_SEGMENT_DEPT_CLASS).

Financial Transactions

ReIM writes two types of transactions to the financial staging tables. Documents which are expected to be sent to the A/P system are sent to the IM_AP_STAGE_HEAD and IM_AP_STAGE_DETAIL tables. Transactions expected to go to the G/L system go to the IM_FINACIALS_STAGE table. The RFI integration will send the data in these tables to EBS or Peoplesoft. Retailers using other financial systems can still use these tables to integrate to their specific A/P and G/L systems.

Complex and Fixed Deal-Related Posting

For complex and fixed deals, batch processes copy most of the data from the RMS staging tables into ReIM detail tables (IM_COMPLEX_DEAL_DETAIL, IM_FIXED_DEAL_DETAIL). Some of the data on these tables is later referenced during the posting process for the created documents, including:

- Location
- Item

Financial Posting

To understand the process that posts data from ReIM to the financials staging table (IM_FINANCIAL_STAGE), see "Financial Posting Batch Design".

Tracking Receipt Posts

Receipt tracking functionality allows the retailer to track what receipts have posted. This processing helps the retailer check the integrity of its financial data.

Note that Oracle Retail does not provide packaged reporting in conjunction with this processing. Rather, the retailer builds its own processes and creates its own reporting mechanisms against the data resulting from the receipt tracking functionality.

Tables Related to Tracking Receipt Posts

In-Process Tables

The tables illustrated below are for the retailer's understanding, but the data on these tables should not be used by the retailer as it builds its processes and reports.

Each area of the system that matches receipts to invoices updates the IM_RECEIPT_ITEM_POSTING table. This table tracks how much of an individual receipt item has been matched and posted.

IM_RECEIPT_ITEM_POSTING

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
RECEIPT_ID	NUMBER(10)	N
ITEM_ID	VARCHAR(25)	N
QTY_MATCHED	NUMBER(12,4)	Y
QTY_POSTED	NUMBER(12,4)	Y

IM_RCPT_ITEM_POSTING_INVOICE

Column Type	Type	Nullable
SEQ_NO (from IM_RECEIPT_ITEM_POSTING)	NUMBER(10)	N
DOC_ID	NUMBER(10)	N
STATUS	VARCHAR2(1)	Y

Staging Tables to be used for Reporting Once posting is completed, the following staging tables contain all currently posted entries. Thus, to build processes and reporting that tracks receipt posts, the retailer should use only the data from these staging tables.

IM_RECEIPT_ITEM_POSTING_STAGE

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
RECEIPT_ID	NUMBER(10)	N
ITEM_ID	VARCHAR(25)	N
QTY_POSTED	NUMBER(12,4)	N
CREATE_DATE	DATE	N

IM_RCPT_ITEM_POSTING_INV_STAGE

Column Type	Type	Nullable
SEQ_NO	NUMBER(10)	N
DOC_NO	NUMBER(10)	N

Multiple Lines for an Individual Receipt Item

For a given line item on a receipt, a line item can be split between multiple invoices. For example, one invoice could match half of a line item; another invoice could match the other half of the line item. Two separate lines would thus appear. The retailer should note that these values (and those in equivalent business scenarios) need adding together to indicate how much of a given receipt item is posted.

LDAP and Other User Interfaces

ReIM supports only LDAP user authentication. To indicate the authentication method, select LDAP for the property called authentication_source in the reim.properties file.

LDAP

Light Directory Access Protocol (LDAP) is the means of user authentication supported by ReIM. It defines a network protocol for accessing information in a directory.

Only LDAP is used within ReIM for user authentication. Because ReIM has specific requirements for ReIM user roles and permissions that are easily configurable by the retailer, they are defined in the application itself. ReIM reads standard user information from an LDAP server.

There are two types of roles; business roles and enterprise roles. Business roles are defined within the ReIM application and define what functional privileges the user has. Enterprise roles are defined in LDAP (it is the same as user group). Any ReIM user defined in LDAP should be part of an LDAP user group defined by roles_base_dn in ldap.properties

If the retailer already stores user information using LDAP, the only interfacing configuration required is through the LDAP-specific properties file. The entries in this file point ReIM to the appropriate machine and port to find the LDAP server. Other properties also may be modified to reflect the names of attributes that the retailer uses in its LDAP schema.

Setting up the LDAP interface entails completing tasks within LDAP andReIM as follows.

Setup Steps within LDAP

In LDAP, complete the following steps.

1. Define the following attributes (or find the existing attributes that provide the same information) within your LDAP (actual name is not important).
   - User ID (standard uid).
   - Password.
   - First name (standard cn).
   - Last name (standard sn).
   - Preferred user language (identifies user locale).
   - Preferred user country (identifies user locale).
   - Preferred email (standard mail).

   Note: All attributes listed above should be single value and mandatory. For attributes with multiple values, the first value is used within ReIM.

2. Create an attribute class that encompasses all the attributes above.
3. Select a container within your LDAP to hold users, which may be created directly in the container or in enclosed containers.
4. Either create new users based on the attribute class just created above, or add the attribute class to the existing users. Define missing values as needed.

Setup Steps within ReIM

In ReIM, complete the following steps.

1. In reim.properties, change authentication_source to LDAP.
2. In ldap.properties, define the values for the parameters shown in the following table.

Parameter	Description
connection_url	Machine and port for your LDAP server
ldap_alias	Alias for the wallet entry containing the credentials for the user defined within LDAP that has ADMIN privileges
base_dn	Name of the container for ReIM users
roles_base_dn	Name of the container for ReIM role
login_id_attribute_name	Name for user ID attribute used within the attribute class
user_container	Name of the container for ReIM users
user_first_name_attribute_name-	Name for the first name attribute used within the attribute class

Parameter	Description
user_last_name_attribute_name	Name for the last name attribute used within the attribute class
user_email_attribute_name-	Name for the email attribute used within the attribute class
user_password_attribute_name-	Name for the password attribute used within the attribute class
user_language_attribute_name	Name for the preferred language attribute used within the attribute class
user_country_attribute_name-	Name for the preferred country attribute used within the attribute class
user_main_key	User attribute which stores the username to be used in ReIM
role_member	Group attribute which stores the distinguished name of users in the group
role_application	Group attribute which stores the group name.
WildCard	Wild character used to identify any search criteria filter
referral_handling	Property defining how the LDAP should handle referrals.

Additional LDAP Resources

- http://www.openldap.org/
  This site contains the OpenLDAP main page. This site contains introduction, downloads, and documentation.
- http://www.iit.edu/~gawojar/ldap/
  This site is the LDAP browser site.
- http://ldap.akbkhome.com/
  This site contains an LDAP schema view with some definitions of the standard LDAP object classes and attributes.

4

Technical Design

This chapter contains information related to the technical design of ReIM.

Locking Design Summary

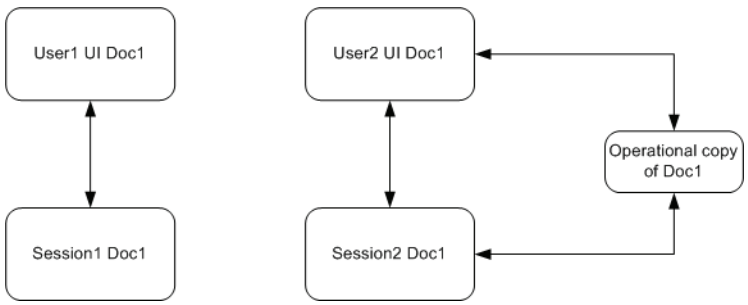
ReIM user interface is based on the optimistic locking strategy. It means that no entities will be preemptively locked before modification. Only at the point when modification needs to be performed the locking mechanism will be used. If two users are trying to modify the same entity, for example the same invoice, both users can do this on the screen. At the same time only the first user to attempt to save the changes will actually successfully save them and the second user will get an error message and would need to refresh the entity and try again.

Example

User1 is trying to update document D0. Document D0 has terms T0. User1 is trying to update T0 to T1. User2 is also trying to update document D0. User2 is trying to update T0 to T2. If User1 will be the first to attempt to save the change to D0 then D0 will have terms T1. User2 would need to refresh D0 to see the ne value of the terms and only after that would be able to change T1 to T2.

All user interface workflows can be sub-divided into 2 categories - simple entity workflows and complex process workflows. For simple entity workflows the locking is handled by the ADF framework. For complex workflows the workflow entities are copied into session (workspace) tables. For such workflows the locking is done by custom logic to allow data to be modified both by ADF framework code and by backend PL/SQL.

To facilitate locking, all tables that hold modifiable data have a standard - OBJECT_VERSION_ID - that is updated every time the entity is modified. New entity is created with OBJECT_VERSION_ID of 1. Modification verification is performed by comparing the copy's being modified version id with the original copy version id. If it different then the user's copy of the entity is out of sync.



In a batch mode no locking is performed. It is assumed that the batches will be run during the batch windows. It means the assumption is that he batch user will be the sole user of the entities. Because of this the batches should not be run ad-hoc during the normal business hours.

Currency Design Summary

ReIM has been designed to handle a multiple number of currencies. This section addresses the system's assumptions, conversion process, and validations that are related to this capability.

Merchandising System (such as RMS) and ReIM Assumptions

Consider the following assumptions.

- RMS defines one currency as the primary currency of the system (held on the RMS SYSTEM_OPTIONS table in the CURRENCY_CODE field).
- RMS specifies that each purchase order can have one currency. This purchase order currency does not have to be the same as the RMS primary system currency or the RMS supplier currency.
- ReIM requires that each document have its currency stated (IM_DOC_ HEAD.CURRENCY_CODE). This invoice currency does not have to be the same as the system primary currency.
- ReIM assumes that a purchase order and any invoices associated with that purchase order are in the same currency. This assumption is based on the business reality that these currencies are almost always the same and on the development consideration that currency conversion processes have an adverse impact on system performance.

Currency Conversion Process for Amount Tolerances

The following is information about the currency conversion process for amount tolerances.

- Amount tolerances are established in the currency of the tolerance entity (each tolerance entity has a currency setting). However, because the invoices and POs to be matched could reflect a different currency, amount tolerances must be converted before they can be applied. In other words, the currency established for amount tolerances is converted when the invoice/PO combination is not in the tolerance currency. For example, a tolerance defined as 10 US dollars (USD) has a much different meaning than a purchase order/invoice defined in Thai Baht (10 Thai Baht is about 0.23 USD). If the system merely utilized the number 10 and failed to perform a currency conversion, the amount tolerances would not apply correctly.
- Currency conversion rates are stored on the RMS CURRENCY_RATES table. The conversion factors on this table are in terms of the primary currency of the system. For example, suppose a retailer wishes to convert from Thai Baht to Uruguayan Pesos and the system's primary currency is USD. First, the system performs a conversion from Thai Baht to USD. Secondly, the system converts the USD value to Uruguayan Pesos. In other words, to perform its conversions, the system always must 'go through' the primary currency of the system.
- Currency conversion is always done based on he consolidated exchange rate.

Currency-Related System Validations

One of the validations performed by the EDI Injector process is that it determines whether the currency on the invoice is the same as the currency on the purchase order. If the invoice currency is not the same as the purchase order currency, the invoice is rejected.

The graphical user interface (GUI) invoice entry process also validates that the currency on the invoice is the same as the currency on the PO associated with the invoice. If the currencies are not the same, the user receives a warning message.

Currency Formatting

Monetary values must be properly formatted according to the associated currency for matting rule. The precision for the currency formatting is defined in RMS CURRENCIES table.

5

System Configuration

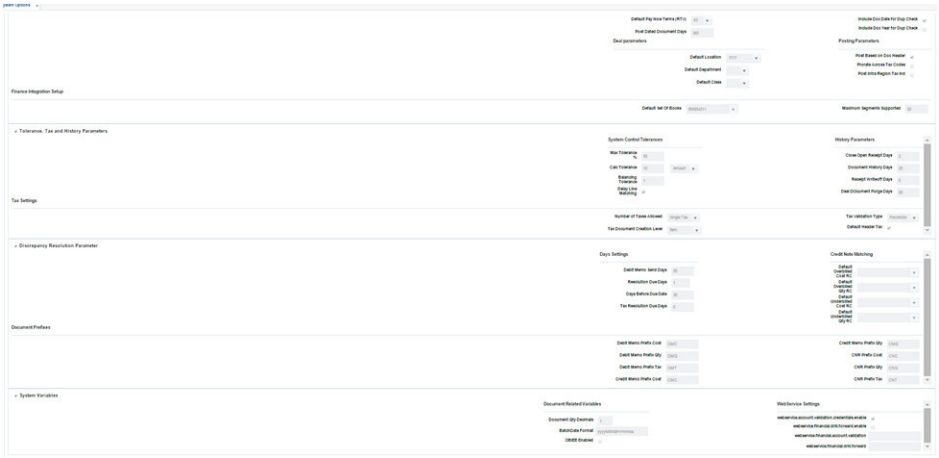
After installation, some of the mandatory steps need to be performed for the application to be functioning correctly. The objective of the checklist is to make sure that all the required data elements have been provided to eliminate failure in the document processing lifecycle.

Define System Options

Out of the box some of the system options have been set to default values. You need to verify that the values meet your business requirements. Some of the attributes that would need to reference identifiers for the existing entities could not be defaulted and would need to be set explicitly. Such attributes are:

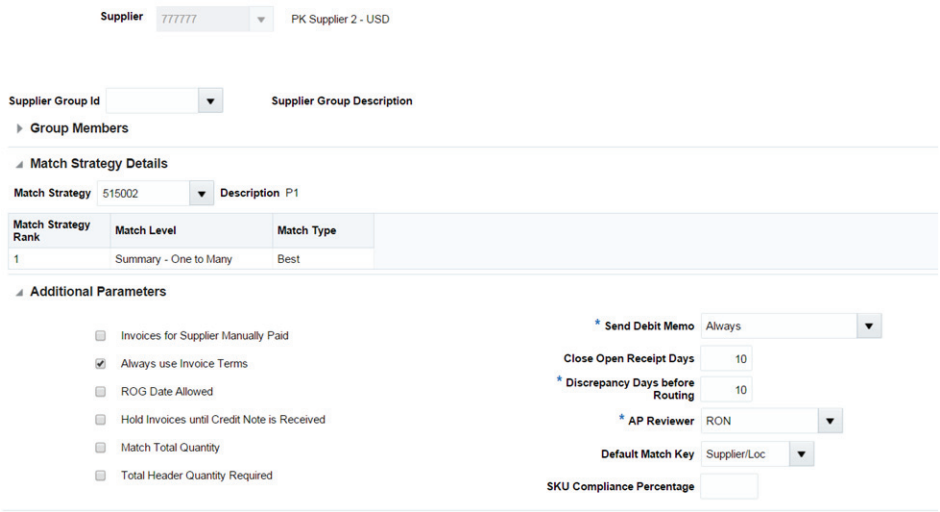
- Default Pay Now Terms (Deals)
- Default Pay Now Terms (RTV)
- Default Location
- Default Department
- Default Class
- Default Set Of Books
- URL for account validation web service
- URL for drill forward web service

Please see the *Oracle Retail Invoice Matching User Guide*, System Options section for more information.



Define Supplier Options

Each supplier in Retail Merchandising System would need to have Supplier Options defined before it can be "visible" within the Invoice Matching application. Supplier options can be defined at the Supplier level, Supplier Site level and Supplier Group level. Only Supplier level options are mandatory.



Please see the *Oracle Retail Invoice Matching User Guide*, Supplier Options section for more information.

Define Matching Tolerance

If you expect to match documents within tolerance rather than matching exactly, you would need to define tolerance entities and map tolerances to appropriate tolerance nodes. Please see the User Guide, Tolerance Maintenance and Tolerance Mapping sections for more information.

Define matching Strategies

If you expect to run Auto Match batch process, and you probably would, you would need to define some Matching Strategies. Please see the User Guide, Match Strategy Maintenance section for more information.

Define Reason Codes

Reason codes would need to be defined to perform discrepancies resolutions. Reason codes are synonyms for the resolution actions that the application user can take to resolve matching discrepancies. This release of Invoice Matching application doesn't provide a dedicated screen for reason code maintenance. As such, the reason codes would need to be defined via back-end. The reason codes are defined in IM_ REASON_CODES table.

An example reason code script to populate the table would look similar to the following:

```
INSERT INTO IM_REASON_CODES ( REASON_CODE_TYPE
                            , REASON_CODE_ID
                            , REASON_CODE_DESC
                            , ACTION
                            , COMMENT_REQUIRED_IND
                            , HINT_COMMENT
                            , DELETE_IND
                            , CREATED_BY
                            , LAST_UPDATED_BY )
 SELECT 'CNT'
 AS     REASON_CODE_TYPE
      , NVL ( ( SELECT MAX ( TO_NUMBER ( REASON_CODE_ID ) ) + 5
                  FROM IM_REASON_CODES )
              , 100 ) AS REASON_CODE_ID
      , 'CN TAXES ARE INCORRECT' AS REASON_CODE_DESC
      , 'CNRTF'                  AS ACTION
      , 'N'                      AS COMMENT_REQUIRED_IND
      , NULL                     AS HINT_COMMENT
      , 'N'                      AS DELETE_IND
      , 'REIM_DEMO_USER'         AS CREATED_BY
      , 'REIM_DEMO_USER'         AS LAST_UPDATED_BY
   FROM DUAL;
```

You can also consult *im_demo_data.sql* script that is shipped with the product. Please note that the script would need to be modified to suite your needs.

There should be at least one entry for each action that would need to be performed. Actions are subdivided based on categories (REASON_CODE_TYPE).

- Cost ('C')
- Quantity ('Q')
- Credit Note Matching related ('CNT')
- Return to vendor ('RTV')
- Tax ('T')

The same action can be mapped to more than one reason code. If you would need some comments to be provided when using the reason code, set the COMMENT_REQUIRED_IND to 'Y'. Hint comment can be optionally provided.

In addition to populating IM_REASON_CODES table with the reason codes, you would also need to populate IM_SEC_GRP_REASON_CODE table for each reason code. This table assigns reason codes to the security user groups defined in RMS. Only the reason codes that have been assigned to the user's security group will be available for processing. There should be a record for each user group/reason code that would require reason code access.

An example reason code security script to populate the table would look like:

```
INSERT
INTO
  IM_SEC_GRP_REASON_CODE
SELECT
  SG.GROUP_ID,
  IRC.reason_code_id
FROM
  im_reason_codes IRC,
  SEC_GROUP SG
WHERE
  IRC.delete_ind  ='N'
AND SG.GROUP_NAME = 'REIM_GROUP';
```

The example is also available in the *im_demo_data.sql* script.

See the *Oracle Retail Invoice Matching Data Model* for more information on the tables structure.

Define GL Mappings

To be able to successfully post transactions to Financial System, such as EBS you would need to define General Ledger account mappings. This release of Invoice Matching application doesn't provide a dedicated screen for General Ledger mappings. As such, the mappings would need to be defined via back-end. The mappings are defined in IM_GL_CROSS_REF table.

An example GL Cross Ref script to populate the table would look similar to the following:

```
INSERT
INTO
  IM_GL_CROSS_REF
  (
    ACCOUNT_TYPE,
    ACCOUNT_CODE,
    SEGMENT_NO,
    SEGMENT_VALUE,
    SET_OF_BOOKS_ID,
    TAX_CODE,
        CREATED_BY,
        LAST_UPDATED_BY
  )
SELECT
  'BT'                AS ACCOUNT_TYPE,
  'TAX'               AS ACCOUNT_CODE,
  10                  AS SEGMENT_NO,
  'BTTAXSEG10'        AS SEGMENT_VALUE,
  FGS.SET_OF_BOOKS_ID AS SET_OF_BOOKS_ID,
  'NOCODE'            AS TAX_CODE,
  'REIM_DEMO_USER'    AS CREATED_BY,
  'REIM_DEMO_USER'    AS LAST_UPDATED_BY
FROM
  FIF_GL_SETUP FGS;
```

The example also is available in im_demo_data.sql script.

The application supports up to 20 segments. Not all segments are required. Usually the number of segments doesn't exceed 10. There should be a separate set of mappings for each set of books defined in the system. Mappings would need to be defined for all basic transactions, as well as for all non-merchandising codes and for all reason codes defined in the previous step.

Optionally, the mappings can be differentiated per tax code.

In addition to populating IM_GL_CROSS_REF table with the GL mappings, you would also need to populate IM_GL_OPTIONS to indicate what segments, if any, would be dynamic. Dynamic segments don't need explicit mappings, but rather the mappings are dynamically determined based of the segment meaning.

An example GL Options script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_GL_OPTIONS
  (
    SEGMENT_NO,
    DYNAMIC_IND,
    SET_OF_BOOKS_ID,
    SEGMENT_LABEL,
    BUSINESS_ATTRIBUTE,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  '10'                AS SEGMENT_NO,
  'N'                 AS DYNAMIC_IND,
  FGS.SET_OF_BOOKS_ID AS SET_OF_BOOKS_ID,
  'Future4'           AS SEGMENT_LABEL,
  'Future4'           AS BUSINESS_ATTRIBUTE,
  'REIM_DEMO_USER'    AS CREATED_BY,
  'REIM_DEMO_USER'    AS LAST_UPDATED_BY
FROM
  FIF_GL_SETUP FGS;
```

The example also is available in im_demo_data.sql script.

If Location segments are defined as dynamic then location segment mapping would need to be provided. It is done by populating IM_DYNAMIC_SEGMENT_LOC table. There should be an entry for each location that would need to be posted. There should be a separate set of mappings for each set of books defined in the system.

An example location mapping script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_DYNAMIC_SEGMENT_LOC
  (
    LOCATION,
    LOC_SEGMENT,
    COMPANY_SEGMENT,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  STORE                          AS LOCATION,
  ('DEMOS10000000' ||STORE)      AS LOC_SEGMENT,
  'DEMO12345678910'              AS COMPANY_SEGMENT,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  STORE
UNION
SELECT
  WH                             AS LOCATION,
  ('DEMOW20000000'|| WH)         AS LOC_SEGMENT,
  'DEMO12345678910'              AS COMPANY_SEGMENT,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  WH
WHERE
  WH = PHYSICAL_WH;
```

If Department or Class segments are defined as dynamic then merchandising hierarchy segment mapping would need to be provided. It is done by populating IM_DYNAMIC_SEGMENT_DEPT_CLASS table. There should be an entry for every department/class combination that would need to be posted. There should be a separate set of mappings for each set of books defined in the system.

An example department/class mapping script to populate the table will look similar to the following:

```
INSERT
INTO
  IM_DYNAMIC_SEGMENT_DEPT_CLASS
  (
    DEPT,
    CLASS,
    DEPT_SEGMENT,
    CLASS_SEGMENT,
    SET_OF_BOOKS_ID,
    CREATED_BY,
    LAST_UPDATED_BY
  )
SELECT
  C.DEPT                                AS DEPT,
  C.CLASS                               AS CLASS,
  ('DEMOD33333333' || C.DEPT)           AS DEPT_SEGMENT,
  ('DEMOC44444444' || C.CLASS)          AS CLASS_SEGMENT,
  FGS.SET_OF_BOOKS_ID                   AS SET_OF_BOOKS_ID,
  'REIM_DEMO_USER',
  'REIM_DEMO_USER'
FROM
  CLASS C, FIF_GL_SETUP FGS;
```

See the *Oracle Retail Invoice Matching Data Model* for more information on the tables structure.

6

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 a shared processing with the UI in a client server model.

Services retrieve the data on which the batch processes work to complete their tasks. The service layer consists of a collection of Java classes that implements business logic (data retrieval, updates, deletions, and so on) through 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. However, the volume can be managed using the inclusion-exclusion configuration feature (see section Batch Configuration)
- ReIM batches should run only in the batch window when no users exist in the system. This requirement is related to locking consideration (see locking section for more information).
- They run in a client server model. Client side code is a plain java which basically responsible for the validation of input parameters, retrieval of user credentials for the given alias name.
- In case there is no valid user/password pair is found for the given alias name, the client program complains the same and terminates. On the successful identification of credentials, client program makes a call to execute<<Batch>> method of corresponding EJB on the application server

Batch Process Configuration

For programs in the financial posting batch cycle, we have implemented a new configuration feature. The volume of data processed by each batch execution can be controlled through a table configuration. While executing a particular batch program (say AUTOMATCH), we may restrict the batch candidates to either a particular supplier or exclude a particular supplier to manage the batch load. At any point of time, a max of one exclude or one include should be configured per a particular day.

A new table IM_BATCH_CONFIG is included in the data model which is used for configuring the exclusion or inclusion of suppliers per batch for a particular day. An indicator called 'PROCESSED_IND' is in the table that tells if the inclusion/ exclusion are already processed or not. Its initial value is N.

Table 6–1 IM_BATCH_CONFIG Parameters

Parameter	Description
BATCH_NAME	Used for configuring the batch name.
SUPPLIER	The supplier that the batch should exclude/include.
SCOPE	Exclude (E)/Include (I)
PROCESSED	Initial value N. Once the batch completes, program sets it to Y so that it would not considered next execution.
PROCESSED_DATE	This attribute is added for the auditing purpose. It also cater as a constraint to make sure that only one Exclude or Include are configured per supplier per day.

The query that picks the candidates for the each batch is modified to join with IM_ BATCH_CONFIG table to first exclude the documents/data that are of SUPPLIERS configured to exclude in the particular batch program with a PROCESSED value N. Then the inclusion logic is applied, if any. If no exclusions/inclusions have been configured for a particular batch with PROCESSED value N, then the batch will process all the eligible data.

The batches that have this capability are

- Automatch batch
- Financial Posting batch
- Resolution Action Rollup batch
- Complex Deal Upload
- Fixed Deal Upload

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 Injector) 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 Injector and the EDI invoice download batch processes are described later in this chapter. For the EDI Injector batch, the input file/folder and the rejection file/folder should be on the same physical machine as the application server is running on. This is a requirement from the file system access permission stand point. Similarly, the EDI download batch process can save the edi output file only to a folder that is on the same machine as the server is running on.

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:

- Auto-match
- Batch purge
- Account purge
- Reason Code action rollup

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:

- Complex deal upload
- Fixed deal upload

Batch Names

The following table describes ReIM batch processes. The table 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 with which it interacts.

Table 6–2 Batch Names

Batch Name	Class (oracle.retail.apps.reim.batch.client)
Tables purge	TablesPurgeBatchClient
Account purge	AccountWorkspacePurgeBatchClient
EDI Injector	EdiInjectorBatchClient
Auto-match	AutoMatchBatchClient
Receipt write-off	ReceiptWriteOffBatchClient
Reason code action rollup	ReasonCodeActionRollupBatchClient

Table 6–2 (Cont.) Batch Names

Batch Name	Class (oracle.retail.apps.reim.batch.client)
Financial posting	FinancialPostingBatchClient
EDI Invoice download	EdiDownloadBatchClient
Complex deal upload	ComplexDealUploadBatchClient
Fixed deal upload	FixedDealUploadBatchClient
Financial Posting workspace Purge	FinancialPostingWorkspacePurgeBatchClient

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:

Table 6–3 ReIM Batch Processes

Batch Processes	Details	Batch Dependencies
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).	
Account purge	This process deletes the accounts maintained locally in the ReIM application.	
EDI Injector	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.	▪ EDI Injector ▪ Receipt upload (Merchandising system, such as RMS)
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.

Table 6–3 (Cont.) ReIM Batch Processes

Batch Processes	Details	Batch Dependencies
Reason code action rollup	This batch process sweeps the action staging table and creates debit memos, credit memos, and credit note requests 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. If hold invoice functionality is on, each generated document is assigned the invoice number to which it corresponds to ensure all related documents are released to accounts payable at the same time. 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 rolled up for an invoice only if no invoice lines on the invoice have any pending adjustments.	
Financial posting	A recurring resolution posting process retrieves all matched invoices and approved documents. If hold invoice functionality is used, then matched Credit Notes rather than approved Credit Notes are processed. Documents which should be sent to the Financial A/P system are sent to the AP staging tables: IM_AP_STAGE_HEAD, and IM_AP_STAGE_DETAIL. Transactions which should be sent to the Financial G/L system are sent to the IM_FINANCIALS_STAGE table	

Table 6–3 (Cont.) ReIM Batch Processes

Batch Processes	Details	Batch Dependencies
EDI invoice download	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. 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 through the EDI invoice download transaction set.	Auto-match must run prior to the EDI invoice download.
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.	
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.	

Features of the Batch Processes

This section describes the features of 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

Batch Log and Error File Paths

The client side log file location is determined by the retailer through the logj4.properties file. The errors that occur on the server side program will be written to the server log which can be configured by WebLogic administrator. If an error occurs that causes a batch process to suddenly come to a complete halt, the system writes to the configured log 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 configuration related to some of the multi threaded batches can be configured in REIM System options. 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 constitutes a thread.

Fixed Deal Upload (FixedDealUploadBatch)

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

EDI Injector (EdiInjectorBatch)

This process is threaded into groups of documents. Each thread handles the business validation of the entire document group.

Auto-Match (AutoMatchBatch)

Auto-match is threaded based on the number of invoice-items and receipt-items involved in the match.

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 injector (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

Batch processes are executed through the batch client framework. This framework is responsible for ensuring that the batch job is passed the appropriate arguments. The arguments for the batch runner are as follows:

- Batch job class name
- batch-alias-name
- Batch arguments

> **Note:** Batches are run with an alias name rather than with a user name/password combination. The alias name is mapped to the user credentials inside a password store called a wallet.
>
> At run time the batches access the wallet and retrieve the user ID and password for authentication purposes.

Below is an example of how the batch runner would be utilized to execute the EdiInjectorBatch process:

```
reimediinjector batch-alias-name /dir/input.dat /dir2/output.dat
```

The batch client programs require the application libraries (JAR files) to be on the classpath in order to execute successfully.

Tables Purge Batch Design

The batch purging 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). The TablesPurge 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 TablesPurgeBatch process:

```
reimpurge batch-alias-name [PURGE_OPERATIONALS | PURGE_WORKSPACES | PURGE_
WORKSPACES_AND_OPERATIONALS]
```

The first argument is batch alias name. The second argument is the purge action. PURGE_OPERATIONALS would make the batch purge data from operational tables alone, PURGE_WORKSPACES would make the batch purge data from all workspace tables, PURGE_WORKSPACES_AND_OPERATIONALS would make the batch purge data from both operational and workspace tables. By default, the purge action is committed.

Purge Operational

Data from the operational tables will be purged based on the document history days system option. The tables are purged in the reverse order of their relationship to each other, history tables first and then the details and then the main or primary table.

Purge Workspace

The workspace tables used for display or internal calculation purposes for Search, document maintenance, Matching or Posting processes would be truncated. Indexes on all affected tables would be rebuilt after truncate. It is recommended that the workspace tables be truncated frequently, as they need to be well maintained to ensure optimal performance of the front end.

Purge Workspace and Operational

Data from the workspace tables are truncated followed by the deletion of data from the operational tables.

Primary Tables Involved

The following lists include the tables on which the purging algorithm is applied:

Operational

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_RESOLUTION_ACTION

- IM_RECEIPT_ITEM_POSTING_%
- IM_%_MATCH_HISTORY

Workspace

- IM_%_SEARCH_WS
- IM_MATCH_%_WS
- IM_MATCH_POOL_%
- IM_POSTING_DOC_%

Other tables of less significance also get purged.

Accounts Purge Batch Design

This process deletes the accounts maintained locally in the ReIM application. The batch retrieves the accounts in IM_VALID_ACCOUNTS table and validates the account against the integrated financial system. Accounts that are invalid in the financial system are deleted from IM_VALID_ACCOUNTS table.

> **Note:** Run the batch whenever account information changes are communicated to ReIM.

Usage

The following arguments are applicable for the AccountWorkspacePurgeBatch process:

```
reimaccountworkspacepurge batch-alias-name
```

Major Modules

AccountWorkspacePurgeBatchClient

Major Tables

IM_VALID_ACCOUNTS

EDI Invoice Injector Batch Design

The EDI Injector 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 can cause invalid EDI transaction to be fixable, where the data can be corrected through online process. The rest of the data validation errors cause the invalid transaction to be written to a set of reject files where a user must correct the problems and re-run the files.
- Validates the data against the ReIM system and the merchandising system (RMS).

- Any errors found are recorded in to the error table (IM_INJECT_DOC_ERROR) so that users can audit and fix any transactions that were rejected.
- Adds the data to the ReIM system. All valid transactions are written to the IM_DOC_xxx, IM_INVOICE_xxxtables.
- The size of the Logical Unit of work for each chunk needs to be defined in ReIM's system options.

Usage

The following arguments are applicable for the EDI Injector Batch process:

```
reimediinjector batch-alias-name input-file/input-path output-file/output-path
```

Assumptions and Scheduling Notes

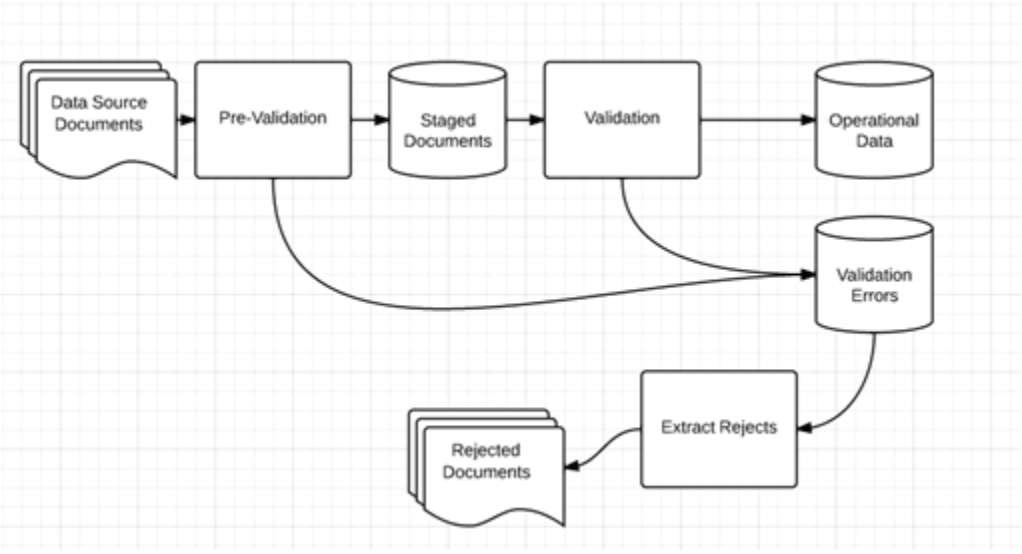
This process can be run ad-hoc but in general it should be run before the auto-match process.

Restart and Recovery

If the EDI Injector Batch aborts without processing an entire file, the file can 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.

High-Level Flow Diagram

The following diagram offers a high-level view of the processing logic utilized within the EDI Injector Batch process.



Primary Tables Involved

The following tables are involved in the EDI Injector Batch process:

Operational Data Tables

- IM_DOC_HEAD
- IM_DOC_TAX
- IM_INVOICE_DETAIL
- IM_INVOICE_DETAIL_TAX
- IM_INVOICE_DETAIL _ALLOWANCE
- IM_INVOICE_DETAIL_ALLW_TAX
- IM_DOC_NON_MERCH
- IM_DOC_NON_MERCH_TAX
- IM_DOC_DETAIL_REASON_CODES
- IM_DOC_DETAIL_RC_TAX

Injector Workspace Tables

- IM_INJECT_DOC_DETAIL
- IM_INJECT_DOC_DETAIL_ALLOWANCE
- IM_INJECT_DOC_DETAIL_ALLOW_TAX
- IM_INJECT_DOC_DETAIL_TAX
- IM_INJECT_DOC_ERROR
- IM_INJECT_DOC_HEADER
- IM_INJECT_DOC_NON_MERCH
- IM_INJECT_DOC_NON_MERCH_TAX
- IM_INJECT_DOC_RECORD
- IM_INJECT_DOC_RULE
- IM_INJECT_DOC_TAX
- IM_INJECT_STATUS

Invoice 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 inputs into the auto-match process include the following:

- Invoices
- Receipts
- Purchase orders
- Match Strategy
- Tolerance

ReIM owns invoices, Match Strategy, and Tolerance while receipts and purchase orders are owned by a merchandising system, such as RMS.

The Match Strategy rules feature allows retailers to build and maintain match strategies which specifically define the types of matches which should be attempted

and the order in which they should be tried during the auto-match process. The match strategies can be defined at the system, supplier group or supplier level. The creation of a Supplier Group is tightly integrated to the logic for selecting documents to be processed by the match engine. If a Supplier Group is created, all the documents for all the suppliers in the group are considered by the match engine together. If a match strategy is defined at the Supplier Group level, then it is used to determine what match attempts to apply against the documents in the supplier group. If a match strategy is not defined at the supplier group level, then the system default match strategy is used to determine which match attempts are used to attempt to match documents for the supplier group. If a match strategy was set up for one of the suppliers for a supplier group, it is ignored by the match engine. If a supplier is not part of a supplier group, then all the documents for that supplier are considered by the match engine together. If a match strategy is defined at the Supplier level, then it is used to determine what match attempts to apply against the documents for that supplier. If a match strategy is not defined at the supplier level, then the system default match strategy is used to determine which match attempts should be used to attempt to match documents for the supplier.

The auto-match process attempts to match the invoices to receipts to the best of its abilities. The process assign 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 if Cost Pre-matching is opted while running the auto-match batch.

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

The size of the Logical Unit of work for each chunk needs to be defined in ReIM's system options.

Usage

The following arguments are applicable for the Invoice Auto-Match Batch process:

```
reimautomatch batch-alias-name
```

Algorithms

The following algorithms comprise the auto-match process:

Cost pre-matching

The Cost Pre-Matching routine is optional but if it is run, it runs as the first step of the Auto-match batch. When the Cost Pre-Matching routine is run, it is run against all suppliers. The routine is only executed if no receipt exists for the order on the invoice. If it finds differences in the cost on the order and the cost on the invoice which are outside of the tolerance level, it generates a cost discrepancy. 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.

Summary matching

The Auto match batch attempts various types of Summary Matches based on the Match Strategy associated with the supplier or supplier group. Summary Matching involves looking at the total document values (cost and optionally quantity) without considering the specific items on each document.

Summary Match All-to-All

The all-to-all match attempts to match all invoice documents to all receipt documents in the match pool. Used in combination with the Match Strategy table, all-to-all matching provides the user:

- The option to choose whether or not to run the all-to-all match.
- The option to choose the order in which the all-to-all match is attempted. The user could decide to execute other match attempts before the all-to-all match.
- The option to decide how to group the invoices and receipts together to attempt matching by specifying the match key.

Summary Match One-to-Many

The one-to-many match attempts to match one invoice document to one or more receipt documents. There are two options when performing a one-to-many summary match:

- Regular Match

  Regular Match attempts to match the invoices and receipts in the pool as one to one matches. If an invoice could match to two or more receipts within tolerance, then the match fails. Similarly, if two or more receipts could match to a single receipt (within tolerance), then the match fails and both invoices are put in multi-unresolved status. If Regular Match fails because the invoice could be matched to multiple receipts or if it failed because multiple invoices could be matched to one receipt, the invoice is flagged as multi-unresolved. If a Regular Match fails for any other reason, the invoice is flagged as an 'Unresolved' match.

- Best Match

  The Best Match setting applies additional logic to select better matches when multiple receipts or receipt combinations can be matched to a single invoice. The best match process creates all combinations of one invoice to one or more receipts and selects the best match. The best match logic selects the receipt or combination of receipts that provides the lowest absolute variance. If two potential matches to the invoice have the same absolute variance but one is an overbill and one is an underbill, the underbill takes precedence. If the two potential matches have identical variances, then the invoice quantity matching will be used as an additional criteria. The match with the smallest absolute quantity variance, taken as the best match. If the absolute quantity variances are the same, then no best match can be determined, and the invoice is left as unmatched.

SKU Compliance on Summary Match

The SKU compliance feature can be used to match invoice items and values to what was actually received. SKU compliance is only calculated if all invoices in a match have details. Therefore, if any of the invoices in the match is a header only invoice, the SKU compliance is skipped. SKU compliance checks for how many of the items on the invoice(s) are on the receipt(s) and how many of the items on the receipt(s) are on the invoice(s). There is a percent calculation for each of these ratios, and both ratios must pass the SKU compliance percent for the match to be accepted.

Tax Validation on Header only Matches

ReIM uses a routine in the auto-matching program to perform a tax validation for header only invoices. The tax validation is executed when a header only invoice matches (either perfectly or within tolerance). The tax validation compares the taxes on the invoice to the taxes generated by the items from the receipt. In addition, the tax validation:

- ensures that all tax codes used on the invoice(s) are also used on the receipts in the match, and that the tax rates are exactly the same.
- ensures that all tax codes on the receipts used in the match are also on the invoice(s) and that the rates match.

If the match passes these two criteria, the invoice (and receipts) can be considered matched. If the validation fails, the invoice(s) are put into tax discrepant status.

Detail matching

In auto-matching, matching can be performed for entire invoices or broken down to the line level. Detail matching is performed by item.

- Eligibility for Detail Matching

  In order to be eligible for detail matching, an invoice or receipt must meet the following conditions:

  1. Item lines must be present on the invoice:
  2. The invoice or receipt must not be part of a manual group.

- Regular versus Best Match

  Regular detail matching compares the invoice item with the matching receipt item from all receipts in the pool. When regular matching is only done within a PO, the unit cost on all the receipt items is the same. If the match key being used allows the user to cross PO's, a constraint is included to require all receipt costs on an item to be the same. If receipt costs are different for the same item, detail matching is not allowed for the item.

  The Best Match Strategy for detail matching does two separate routines to attempt to match items within the match pool:

  1. If the cost of the item on all invoices in the pool is the same, Best Match attempts to match all invoices to all receipts in the pool for that item (an all-to-all match). If they are within tolerance for both cost and quantity in the all-to-all step, then the item is matched on all invoices and on all receipts within the pool. If either the cost or quantity match fails, then nothing is flagged as matched.
  2. For the item, look at the item on each invoice in the pool individually and compare it to the sum of all receipts in the pool and select the best match. The criteria for determining the best match in this scenario is as follows:
     - Calculate the unit cost variance, and if it is out of tolerance the invoice is rejected from best match consideration.
     - Calculate the quantity variance and if it is out of tolerance the invoice is rejected from best match consideration.
     - Calculate the variance on the extended cost between the invoice item and the receipt item(s). This variance is compared against all matches which pass the previous steps. Compare the absolute variance for all the matches which are eligible for best match consideration. Take the match with the least absolute variance as the best match. If two matches have the same absolute variance but one is an overbill and one is an underbill, select the underbill as the best match. If the variances are identical, then a best match is not possible, so the match is skipped.

  If the Best Match attempt is unsuccessful, it means that the Regular Match would also have not been successful. However, if the routing date has passed you should

attempt regular matching including the auto-resolution process and the generation of discrepancies.

The Regular Match attempts to match the invoice item with the receipt items from all receipts in the match key (receipt unit cost must all match the unit cost of the item on the PO(/loc) for the invoice being matched).

Generating Discrepancies

During regular detail matching, the auto-matching process generates discrepancies for cost and quantity discrepancies which are outside of tolerance.

The Tolerance table includes an Auto Resolution column which is used to determine the variance percent (or amount) allowed to complete an automatic resolution. The Auto Resolution column means that there are three types of discrepancies:

- Discrepancies which are within the 'variance within tolerance' (VWT) setting.
- Discrepancies which have a variance which can be automatically resolved.
- Discrepancies which have a variance which is too large to be resolved automatically. These variances generate a discrepancy and are sent to the Discrepancy Review list.

Automated Discrepancy Resolution

When a discrepancy has been identified as one which can be automatically resolved (based on comparing the variance with the applicable tolerance from the tolerance table), the system looks up the appropriate resolution action on the reason code table using the code assigned on the tolerance table row associated with the discrepancy. The resolution action is applied to resolve the discrepancy. If this is the last item on the invoice to be resolved then the whole invoice is flagged as matched. If this was the last item for the receipt to be resolved, then the receipt is also flagged as matched.

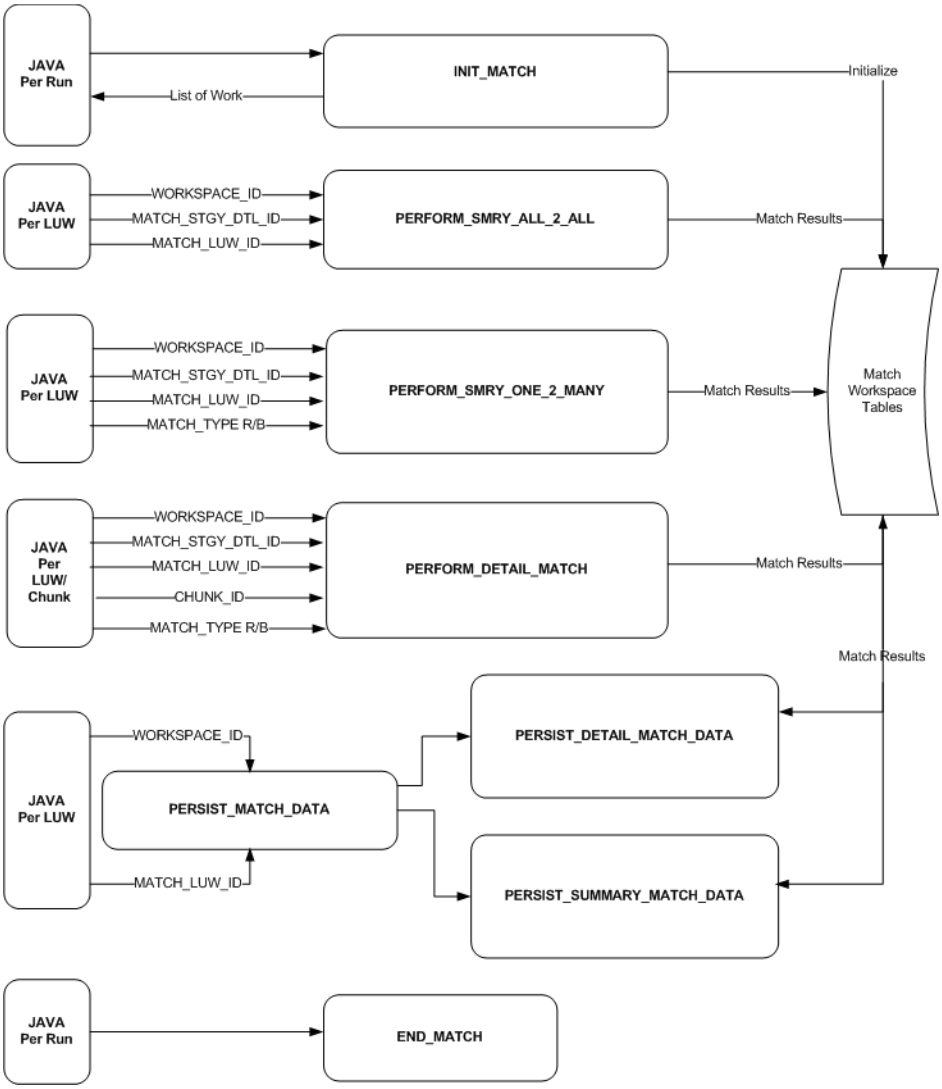
Assumptions and Scheduling Notes

Consider the following assumptions and scheduling notes.

- Auto-match cannot be run during the day when there are users online interacting with the system.
- Both the invoice unit cost and the unit cost of the PO must be expressed in the same currency. In order to compare the invoice unit costs with the PO's unit costs, auto-match does not engage in currency conversion. Match Keys which involve multiple currencies or vat regions or set of books will be removed from the matching process.

High-Level Flow Diagram

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



Primary Tables Involved

The following tables are involved in the Invoice Auto-Match batch process.

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- SHIPMENT(RMS)
- SHIPSKU(RMS)
- IM_PARTIALLY_MATCHED_RECEIPTS
- ORDHEAD(RMS)
- ORDSKU(RMS)
- ORDLOC(RMS)
- IM_TOLERANCE_LEVEL_MAP
- IM_SUPPLIER_OPTIONS
- IM_SYSTEM_OPTIONS

- IM_MATCH_INVC_WS
- IM_MATCH_INVC_DETL_WS
- IM_MATCH_RCPT_WS
- IM_MATCH_RCPT_DETL_WS
- IM_MATCH_GROUP_HEAD_WS
- IM_MATCH_GROUP_INVC_WS
- IM_MATCH_GROUP_RCPT_WS

Credit Note Auto-Match Batch Design

Credit Note Auto-Matching pairs credit note requests to corresponding credit notes sent by the supplier. The CreditNoteAutoMatchBatch attempts auto-matching of credit notes from suppliers, to credit note requests from the retailer without manual intervention. The batch also creates and resolves detail level discrepancies utilizing a predefined set of reason codes. These reason codes are defined within Invoice Matching through the System Options Maintenance screen. In addition, the batch utilizes a variety of configurable keys to allow for document groups to be matched in ways other than just distinct purchase order and location combinations.

When invoked, the batch creates a pool of matchable credit notes and credit note requests. The candidates are selected depending on which customizable fields are populated and a status of credit notes and credit note requests. For information, see the *Oracle Retail Invoice Matching User Guide*.

Once a pool of matchable documents is established, the batch proceeds to group the documents with respect to unique suppliers listed on the documents. Suppliers are the first layer of grouping, which facilitates further processing of each group in parallel using threads.

If threading is enabled for the batch, each supplier based group is processed in its own thread. Each supplier based group further divides the documents for that supplier into smaller document-key sets. These document-key sets are categorized by common attributes defined on the document itself. The attributes, also referred to as Configurable or Flexible 'Pool Keys' allow documents to be grouped in several combinations in addition to the distinct purchase order and location combination (which is the only combination possible in the current Invoice Auto-Matching framework).

Matching is not attempted for groups not containing both credit notes and credit note requests.

By default the CreditNoteAutoMatch process creates document-key sets based on the following key distinctions:

- Deal ID
- Deal Component ID
- Credit Note Request ID
- Original Invoice ID
- PO/Location combination

To enable the use of all five keys, the reference fields in the credit notes and credit note requests must be populated. For information, see the *Oracle Retail Invoice Matching User Guide*. The reference credit note request ID field holds the credit note request ID,

reference invoice ID field holds the original invoice ID, deal ID field holds the deal ID and deal detail ID field holds the deal component ID. In case none of these fields are populated with the required data, the PO/Location combination is the only key available to the CreditNoteAutoMatchBatch process.

Within each document-key set, matching is attempted using three algorithms: summary, one-to-one matching, and detail level matching. Summary-level matching attempts to match all credit notes with credit note requests at a summary level by comparing extended costs, or quantities within tolerance. One-to-one matching requires that extended costs or quantities of one distinct credit note match to only one distinct credit note request within tolerance. Line-level matching is only attempted if there is one unmatched credit note left. It attempts to match the line items of an unmatched credit note with line items of all unmatched credit note requests.

Below is the flow for attempting a match for each of the document-key set:

1. Summary Matching (matching algorithm)
2. One to One Matching (matching algorithm)
3. Line-level Matching (matching algorithm)

If Tax is enabled in the system, CreditNoteAutoMatchBatch only detects Tax discrepancies at the detail level. This means that when documents are being processed by the detail matching algorithm, a check is performed prior to matching, ensuring that the Tax codes and rates for each item on the credit note match those on the credit note request for the corresponding item. When a discrepancy is detected, processing for that document stops and detail matching is not performed for that document. In such a case, the Invoice Matching user will have to match and resolve the Tax discrepancy manually through the user interface.

Tolerances are handled in a manner similar to the Invoice auto-match batch process. The tolerances are first selected with respect to supplier, then with respect to the system. For information, see the *Oracle Retail Invoice Matching User Guide*.

If a match is achieved, the information related to the matched document is migrated to the history tables, and all CreditNoteAutoMatch Batch related tables are purged for those documents. The migration process is enabled depending on the value of the creditnoteautomatchbatch.workspace.cleanup property in the reim.properties file.

In case of an unsuccessful match manual intervention is required through online processes, and the match attempt related data for those documents is not cleaned up from the respective tables. See "Primary Tables Involved" in this section for more details on the tables involved.

Usage

The following arguments are applicable for the Credit Note Auto-Match Batch process:

```
reimcreditnoteautomatch batch-alias-name
```

Algorithms

The Credit NoteAutoMatch batch process includes the following algorithms.

- Summary Matching

  Credit notes and credit note requests in the document set are matched at the summary level by comparing extended costs. If the extended costs of the document set falls within tolerances, the documents are considered matched and flagged as such, processing continues with the next set. Note that since total

extended costs are being compared, only total merchandise amounts will be factored into the actual matching calculations. If the documents in the set are from a supplier that requires quantity matching, quantity matching will be performed within tolerances as well.

- One to One Matching

  One to one matching is a variation of summary matching. It requires that one distinct credit note matches to only one distinct credit note request within tolerance for the document set. Extended costs are compared and quantities are also compared if the supplier option for quantity matching is enabled.

- Detail Matching

  For a given document set, when only one credit note remains unmatched and multiple credit note requests remain unmatched, the system will attempt to match line items from the credit note to the credit note request at the line level. If a match is not found, discrepancies are created and routed for resolution. When discrepancies are created as part of the detail (line-level) matching process, they are automatically resolved by the batch process. This resolution will take place by selecting the appropriate pre-defined reason code from the system options and resolving the discrepancy. During the reason code action rollup process, these newly created resolution actions will be rolled up to create the appropriate resolution documents. In case no applicable reason codes exist in the system for the discrepancy, the credit note will not be matched and processing will stop for the document set.

Assumptions and Scheduling Notes

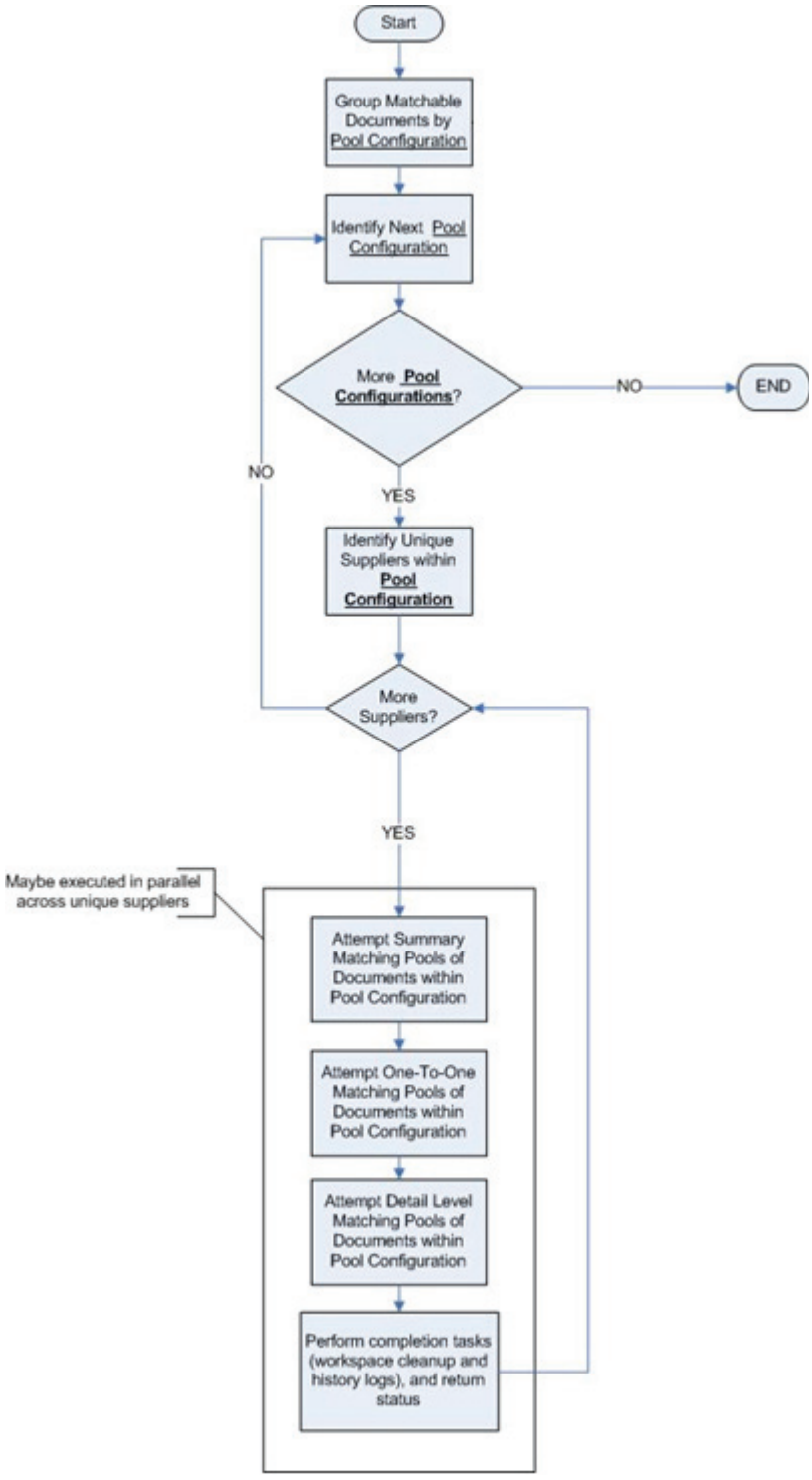
Consider the following assumptions and scheduling notes.

- Both the credit note and credit note request unit cost must be expressed in the same currency.
- The quantities on the credit note must be expressed in the same unit of measure as the quantities on the credit note requests. The batch performs no unit of measure conversion.

Post Processing

- CreditNoteAutoMatch updates the status of qualified documents that have been matched.
- The CreditNoteAutoMatch workspace is cleaned up depending on the related setting in the reim.properties file (refer to the Credit Note AutoMatch Workspace Cleanup Setting section in the reim.properties section).
- The batch creates and resolves discrepancies by utilizing pre-defined reason codes. The Reason Code Rollup Batch must ensure that the respective documents are created.

High-Level Flow Diagram



Primary Tables Involved

The following are lookup tables that must be populated.

Table 6–4

Table Name	Contents
IM_DOC_HEAD	Credit notes and credit note requests with relevant information (such as supplier and status).
IM_SUPPLIER_GROUP_ MEMBERS	Supplier group related information.
IM_DOC_DETAIL_ REASON_CODES	The Item Detail record for credit notes. Data related to items must exist in this table to enable line-level matching.
IM_TOLERANCE_LEVEL_ MAP	Tolerance properties associated with supplier. The data is required when performing matches within tolerances.
IM_SYSTEM OPTIONS	Properties associated with the Invoice Matching function, such as enabling TAX or enabling tolerances.

The following are tables to which the process posts data.

Table 6–5

Table Name	Contents
IM_MATCH_POOL_ CONFIG	Data for the matching process. This data determines which groupings the system utilizes when attempting to match and also dictates the order in which the groupings run.
IM_MATCH_DOC	The pool of documents that the batch process will attempt to match.
IM_MATCH_POOL_ TOLERANCES	The calculated tolerances for each candidate document to be matched.
IM_MATCH_POOL_ RESULTS	Cost and quantity total for a document set being matched and the variance between the documents being matched. Also included is the party the variance favors (retailer or supplier).
IM_MATCH_POOL_ITEM	Actual item detail unit cost and quantities to be used for matching. Details may be from IM_DOC_DETAIL_REASON_ CODES or IM_INVOICE_DETAILS, depending on the type of match performed.
IM_MATCH_QTY_VAR	The quantity discrepancy calculated while attempting a match in a document set.
IM_MATCH_COST_VAR	The cost discrepancy calculated while attempting a match in a document set.

The following new history tables are populated upon the successful completion of the CreditNoteAutoMatch batch. The tables allow the retailer to track match history and locate aggregate data in the other match history tables based on the appropriate match and document type.

Table 6–6

Table Name	Contents
IM_MATCH_DOC_HIST	Upon successful completion of the matching process, documents contained in IM_MATCH_DOC are moved to this history table.
IM_MATCH_POOL_ITEM_ HIST	History of the items that were on the credit note when matched.
IM_MATCH_POOL_ RESULTS_HIST	Data from the MATCH_POOL_RESULTS table is moved to this table after a successful match.

Table 6–6 (Cont.)

Table Name	Contents
IM_MATCH_QTY_VAR_HIST	
IM_MATCH_COST_VAR_HIST	History related to any quantity or cost variance detected during the match process.

The following tables are populated for compatibility with the existing Invoice Matching history maintenance data model.

- IM_CN_SUMMARY_MATCH_HIS
- IM_CN_DETAIL_MATCH_HIS

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 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, 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:

Table 6–7

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.

Usage

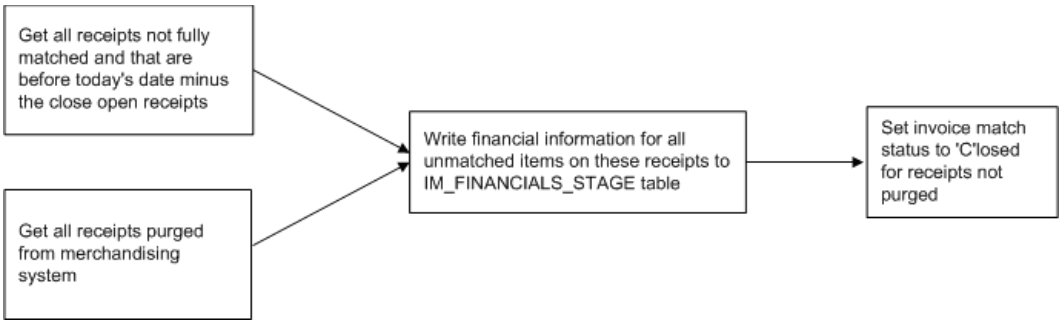
The following arguments are applicable for the Receipt Write-Off Batch process:

```
reimreceiptwriteoff batch-alias-name
```

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

The following tables are involved in the Receipt Write-off batch process.

REIM

- IM_FINANCIALS_STAGE
- IM_SYSTEM_OPTION
- IM_SUPPLIER_OPTIONS
- IM_PARTIALLY_MATCHED_RECEIPTS

RMS

- UNIT_OPTIONS
- SHIPMENT
- STAGE_PURGED_SHIPMENT
- SHIPSKU
- STAGE_PURGE_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.

The action staging table is used during posting to post the reason code actions to the financial staging table.

Usage

The following arguments are applicable for the Reason Code Action Rollup Batch process:

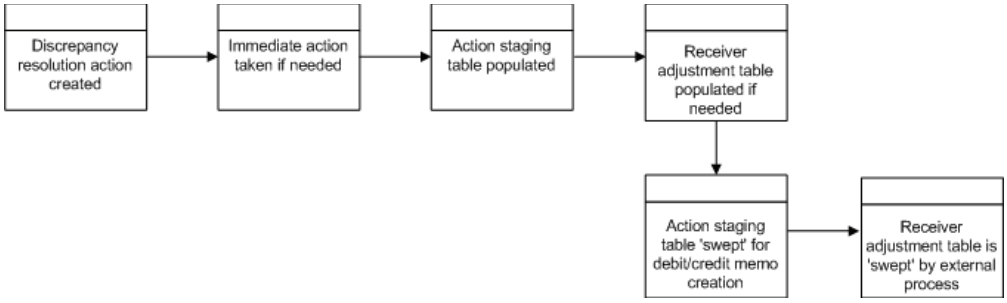
```
reimrollup batch-alias-name
```

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.



Primary Tables Involved

The following tables are involved in the Reason Code Action Rollup batch process.

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

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.

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 moves on to the next invoice/document.

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

Usage

The following arguments are applicable for the Financial Posting Batch process:

```
reimposting batch-alias-name
```

Assumptions and Scheduling Notes

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

- Segment definitions in the GL options.
- GL account segments on the GL Options screen.
- All the accounts using the GL Cross Reference screen.
- 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

The following tables are involved in the Financial Posting batch process.

- The IM_DOC_HEAD table contains the matched, void, and approved documents.
- The IM_DOC_NON_MERCH table contains the non-merchandise costs for invoices.

Lookup Tables that must be Populated

- IM_GL_OPTIONS. Order of segments, business attributes, 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.

Tables to Which the Process Posts Data

Note: The table to which the process posts data is either IM_FINANCIALS_STAGE or IM_AP_STAGE_HEAD

IM_FINANCIALS_STAGE

- Transaction code
- Debit/credit indicator
- Invoice ID
- Invoice date
- Supplier
- Purchase order (if available)
- Shipment/receipt (only if unmatched receipt 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).

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 purposes.
- Doc_id: Similar to IM_FINANCIALS_STAGE.
- The Invoice Type Lookup Code for merchandise invoices and credit memos (where IM_DOC_HEAD.TYPE is MRCHI, CRDMEC or CRDMEQ) is STANDARD. For positive non-merchandise invoices (where IM_DOC_HEAD.TYPE is NMRCHI) the Invoice Type Lookup Code also is Standard. For negative non-merchandise invoices and all other documents, the Invoice Lookup Code is 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.
  - The Org Unit of the Location from this transaction should match the Org Unit of the Site ID. Otherwise, this field value will be null.
- 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.
- Segment 4: Same as in current IM financials staging table.
- Segment 5: Same as in current IM financials staging table.
- Segment 6: Same as in current IM financials staging table.
- Segment 7: Same as in current IM financials staging table.
- Segment 8: Same as in current IM financials staging table.
- Segment 9: Same as in current IM financials staging table.
- Segment 10: 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.

    Recent modifications have been made to the ReIM posting process to better support integration with EBS with respect to VAT requirements. Previous to the modifications, ReIM would post Invoices to the staging tables which passed information to Accounts Payables in a manner where in certain scenarios, the VAT lines could not be easily associated with corresponding Merchandise lines on the invoice. The details of the association between Items

and VAT Codes/Rates is available in ReIM, however it could be lost during the posting process with Accounts Payable.

These new modifications provided a more detailed breakdown of information for items by VAT Code and the association with the appropriate VAT lines. Previous to the modifications, ReIM made it's postings for financial integration by rolling up the Items in the posting to a GL Account Segment level. So, all RMS items that are mapped to the same GL Account Codes in ReIM will be combined into a single posting line. Along with this, TAX lines for the item lines are also posted, one for each VAT Rate that was applicable to the items included in the ITEM line. This did not provide the ability to easily determine the VAT basis that was used to determine the VAT line once posted to the financial system.

The modification changes the level at which the Item lines are posted so that the VAT Rate of the items provides a further breakdown of the Item line posting. The posting now makes the same roll up to the common GL Accounts Segment level and then provides a further breakdown to the VAT code level. TAX line are then posted with each Item line for the corresponding VAT rate of the items. No Item line would have more than 1 TAX line associated with it.

  - 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, where 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.

Financial System Integration

To facilitate the integration process with Financial Systems, a file based integration solution was added to the Financial Posting Batch. An optional flag argument ('Y' or 'N') and extracting location path can be provided to the batch. If specified, the Posting process will extract staged data into the exporting files. The generated CSV files will be produced in the directory specified. There will be up to 3 files produced, one for each staging table:

- IM_AP_STAGE_HEAD
- IM_AP_STAGE_DETAIL

- IM_FINANCIALS_STAGE

Files will have the generation timestamp embedded in the file name. At the end of the extraction process, if the extraction is successful, all staged financial data will be truncated.

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 through the EDI invoice download transaction set to the respective vendors.

Usage

The following arguments are applicable for the EDI Invoice Download Batch process:

```
reimediinvdownload batch-alias-name
```

Assumptions and Scheduling Notes

Consider the following 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.

Usage

The following arguments are applicable for the Complex Deal Upload Batch process:

```
reimcomplexdealupload batch-alias-name block-size partition-no partition-size
```

Primary Tables Involved

Note: For descriptions of RMS tables, see the *Oracle Retail Merchandising System 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_TAX. This table associates the document with its value added tax 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.
- IM_COMPLEX_DEAL_DETAIL_TAX. This table holds the tax information of the complex deal.

Multi-Threading

The Complex Deals upload batch is run in multi-threaded mode as follows:

- reimcomplexdealupload user/password BlockSize PartitionNo

BlockSize

The BlockSize is used to decide how many deal IDs to process in every thread. It should be greater than 1.

For example, if there are 15 deals to be processed in the staging tables and BlockSize input argument is provided as 3, then there will be 5 threads to process 3 deals each simultaneously.

A total of 3 deals records are processed in each of the 5 threads.

PartitionNo

The PartitionNo is used by huge data block that are in the units of millions (for example, 4 million).

The batch is used by the query to pick all the records and it retrieves ALL the deal numbers to be processed by the batch.

For example, the input command line arguments:

reimfixeddealupload user/password 3 1

Generation of Debit Memo (or Credit Note Requests) for Deals

The RMS system generates Debit Memos (or Credit Note Requests) for Fixed or Complex deals and pass them through to ReIM via custom upload batch programs. The RMS system includes a system option called 'Credit Memo Level' which will control the level at which the Debit Memo (or Credit Note Request) is generated.

The valid values for the Credit Memo Level option are:

- L – Location
- T – Transfer Entity
- B – Set of Books
- D – Deal/Component

Therefore, if a retailer wishes to generate a separate Debit Memo (or Credit Note Request) for each location on a deal, they would set the Credit Memo Level option to 'L', and RMS would send a separate transaction for each location on the deal. If this level of detail is not needed, the retailer could set the option to 'D' and only a single document would be sent to ReIM for each deal/component. Note that if the deal/component level is selected, ReIM still internally tracks the locations on each deal and will use this information to credit the correct locations on the deal when making the accounting entries.

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.

For non-merchandise fixed deals that are not associated with an RMS location, the org unit has been added to the RMS staging table. During the Fixed Deal upload process, the set of books ID associated with this org unit is used to access a new table (FIXED_DEAL_SOB_LOC_DEFAULT) to get the location to use for the deal document in IM_DOC_HEAD. Then, the resolution posting job populates the financial staging tables with the set of books ID associated with the location just like it does with all other documents.

Usage

The following arguments are applicable for the Fixed Deal Upload Batch process:

```
reimfixeddealupload batch-alias-name block-size partition-no partition-size
```

Primary Tables Involved

> **Note:** For descriptions of RMS tables, see the *Oracle Retail Merchandising System 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_TAX. This table associates the document with its value added tax 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.
- IM_FIXED_DEAL_DETAIL_TAX. This table holds the tax information of the fixed deal.

Multi-Threading

The Fixed Deals upload batch is run in multi-threaded mode as follows:

- reimfixeddealupload user/password BlockSize PartitionNo

BlockSize

The BlockSize is used to decide how many deal IDs to process in every thread. It should be greater than 1.

For example, if there are 15 deals to be processed in the staging tables and BlockSize input argument is provided as 3, then there will be 5 threads to process 3 deals each simultaneously.

A total of 3 deals records are processed in each of the 5 threads.

PartitionNo

The PartitionNo is used by huge data block that are in the units of millions (for example, 4 million).

The batch is used by the query to pick all the records and it retrieves ALL the deal numbers to be processed by the batch.

For example, the input command line arguments:

reimfixeddealupload user/password 3 1

Generation of Debit Memo (or Credit Note Requests) for Deals

The RMS system generates Debit Memos (or Credit Note Requests) for Fixed or Complex deals and pass them through to ReIM via custom upload batch programs. The RMS system includes a system option called 'Credit Memo Level' which will control the level at which the Debit Memo (or Credit Note Request) is generated.

The valid values for the Credit Memo Level option are:

- L – Location
- T – Transfer Entity
- B – Set of Books
- D – Deal/Component

Therefore, if a retailer wishes to generate a separate Debit Memo (or Credit Note Request) for each location on a deal, they would set the Credit Memo Level option to 'L', and RMS would send a separate transaction for each location on the deal. If this level of detail is not needed, the retailer could set the option to 'D' and only a single document would be sent to ReIM for each deal/component. Note that if the deal/component level is selected, ReIM still internally tracks the locations on each deal and will use this information to credit the correct locations on the deal when making the accounting entries.