Oracle® Invoice Matching

Operations Guide

Release 14.1

E57809-04

March 2016

ORACLE®

Oracle Retail Invoice Matching Operations Guide, Release 14.1

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

5 Integration

6 Technical Design

7 Oracle E-Business Suite Financials Integration using Oracle Retail Financial Integration

8 PeopleSoft Financials Integration using Oracle Retail Financial Integration

9 Batch Processes

Send Us Your Comments

Oracle Retail Invoice Matching Operations Guide, Release 14.1

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.

Related Documents

For more information, see the following documents in the Oracle Retail 14.1 documentation set:

- *Oracle Retail Invoice Matching Release Notes*
- *Oracle Retail Invoice Matching Installation Guide*
- *Oracle Retail Invoice Matching User Guide*
- *Oracle Retail Invoice Matching Data Model*
- *Oracle Retail Merchandising Batch Schedule*
- *Oracle Retail Merchandising Implementation Guide*

- *Oracle Retail Merchandising Security Guide*

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, 14.1) or a later patch release (for example, 14.1.1). 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.

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.

Technical Architecture Overview

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

For more information, see Chapter 3, "Technical Architecture."

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 assumes that the unit of measure (UOM) on the invoice and the UOM on the receipt are always the same and equal to the cost unit of measure (CUOM). (Unit of measure is not displayed on the invoice nor on the receipt.)

- ReIM expects all invoices to be in eaches or the standard unit of measure (SUOM) converted to eaches. No other units of measure can be invoiced using ReIM.

- 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.INVC_MATCH_IND) is set to no. Instead, non-merchandise codes should be added to the NON_MERCH_CODE_HEAD table using the database.

- A record must be inserted into the IM_SYSTEM_OPTIONS table in order to allow successful login to the application.

- 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.

- Multiview

  The Document Find, Group Entry List, and Group Entry pages allow the retailer to define how certain fields display in these screens. The Multiview functionality allows the user to move fields around on the pages and save those views for future use. In order for Multiview to work and for these screens to populate correctly, IM_GLOBAL_PREFERENCES must be populated.

- 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

Retailer-defined configurations for ReIM are located in the reim.properties file. Every setting in the reim.properties file is configurable, according to the retailer's specific business requirements. Some of these settings also are discussed in the Internationalization section later in this chapter.

The key system parameters contained in this file are described in the tables below. Although default values are given in some instances, retailers are responsible for setting these fields appropriately for their installation and hardware profile, rather than assuming these default values are the best choice.

Data Source Configuration Settings

These settings are dependent on the retailer's unique database installation, except for the bean driver, which should remain at the default value (unless customization is performed.)

Table 2–1 Data Source Configuration Settings

Parameter	Description	Comments
datasource.url	The URL connection string used to connect to the database	
datasource.schema.owner	The owning schema used to resolve database types.	
datasource.bean.driver	The bean driver for this installation. This should not change unless customization has been performed.	Default is com.retek.reim.foundation.rms12.
datasource.credential.alias	The alias for data source credentials.	For example, USERALIAS.

Data Source Connection Pool Configuration Settings

These settings are dependent on the retailer's implementation. The pool size refers to the number of available database connections that the retailer intends to keep available.

Table 2–2 Data Source Connection Pool Configuration Settings

Parameter	Description	Comments
pool.name		Default is reim.
pool.min	The least number of database connections available, based on anticipated number of users.	Default is 20.
pool.max	The highest number of database connections available, based on anticipated number of users.	Default is 200.
pool.connectionWait Timeout		Default is 5.
pool.propertyCheckInterval	Sets the time interval. The cache daemon thread "sleeps" between checks to enforce the timeout limits.	Default is 900 (seconds).
pool.timeToLiveTimeout	Sets the minimum time, in seconds. A checked-out connection can remain outside of the cache before it becomes a candidate to be closed by the connection cache thread.	For example, 1200. Default is 0, which disables timeout.
pool.inactivityTimeout	Sets the minimum time, in seconds. A connection can remain idle before it becomes a candidate to be closed by the connection cache thread when the current cache size is greater than the pool minimum.	For example, 600. Default is 0, which disables timeout.
pool.abandonedConnection Timeout	Sets the minimum time, in seconds. A checked-out connection can remain unused (no SQL activity) before it becomes a candidate to be closed by the connection cache thread.	For example, 900. Default is 0, which disables timeout.

Document Settings

Table 2–3 Document Settings

Parameter	Description	Comments
document.search.records.maximum	The maximum number of documents to be returned by the document search screens. This value will depend on the hardware profile. (Integer)	Default is 10000.
document.quantity.decimals.allowed	The decimal precision to which quantity is stored on a document. Typically used by grocery retailers. Value is expressed as an integer. For example, to display 4 decimals, this property is set to 4.	Default is 0.
document.batch.date.format	The date format used for processing and validating EDI files. (String)	Default format is yyyyMMddHHmmss.
document.header.quantity.required	For documents that contain a supplier that does not belong to a supplier group, the value of this field dictates if the total header quantity is required on that document. (Integer)	Default is true.
document.rtv.extdocid.separator	Character used to differentiate similar external document IDs for EDI upload. It allows documents with external document IDs already in use to pass EDI validation. Value must be a valid invoice number character, as defined by the parameter, document.validation.regexpr. If not, the underscore character is defaulted. (Character)	If this property is set to 1, for example, a posted document with an external ID of MYDOC will be named MYDOC_1. Default is _.
document.validation.regexpr	Indicates the characters allowed in an external document (invoice) ID. For example, 0-9, A-Z, space, minus sign, plus sign and underscore. If this property is omitted, the system defaults the setting to alphanumeric. The document.rtv.extdocid.separator value is validated against this field. (String)	For example, ^[0-9A-Za-z\ \+\-_]+$ Default is alphanumeric, expressed as \\p{Alnum}+
document.number.allow.leading.zero	Indicates whether document IDs may be entered with a leading zero.	Valid values: true - document IDs may be entered with a leading 0. false - document IDs may NOT be entered with a leading 0.
document.purge.deals.days	Indicates when to purge deals from the Invoice Matching tables after they have been successfully uploaded (in number of days). This field is read during the purge batch.	Default is 10.
INCLUDE_DOC_DATE_FOR_DUPLICATE_DOC_CHECK	Indicates whether to include document date for unique invoice validation. If true, validation will be based on vendor, document type, invoice number, and invoice date.	Default is false.

Table 2–3 (Cont.) Document Settings

Parameter	Description	Comments
INCLUDE_DOC_YEAR_FOR_DUPLICATE_DOC_CHECK	Used in conjunction with INCLUDE_DOC_DATE_FOR_DUPLICATE_DOC_CHECK to determine unique invoice validation. Only taken into account if INCLUDE_DOC_DATE_FOR_DUPLICATE_DOC_CHECK is set to true. If both are true, then validation will be based on vendor, document type, invoice number, and invoice year.	Default is false.
always.show.suppliersite	Sets whether the supplier site will always be shown when creating a document	Default is true.
intra_region_post_ind	Indicates whether to post tax for intra region documents.	Default is false.

Date Setting

Table 2–4 Date Setting

Parameter	Description	Comments
date.cache.poll.interval	This parameter dictates how frequently the system updates the stored VDATE. (Integer)	Default is 15 minutes, expressed as 900000. (1000 * 60 * 15 = 900000)

Array Process Size Settings

This setting establishes the size of the batch updates to the database. The value is expressed in number of records.

Note: The EDI Injector Batch process does not use this property.

Table 2–5 Array Process Size Settings

Parameter	Description	Comments
ARRAY_PROCESS_SIZE	The threshold representing the maximum number of records to be part of a single INSERT or UPDATE operation in classes created by the DALGenerator. (Integer)	Default is 30.

EDI Properties

Table 2–6 EDI Properties

Parameter	Description	Comments
edi.data.generator.path	The path used for the data generator files. (Ignore in production.) (String: @deploy.data.path@)	
edi.default.location	The default location assigned to non-merchandise invoices when they come in. (Integer)	Default is -.
edi.default.department	The default department assigned to non-merchandise invoices when they come in	Default is -.
edi.default.class	The default class assigned to non-merchandise invoices when they come in	Default is -.
edi.upload.multithreaded	Indicates whether EDI uploads are single-threaded (False) or multi-threaded (True).	Default is true.
edi.docbulk.size	The maximum number of documents that EDI will process before issuing an INSERT statement.	Default is 1000.

Lookout Timeout Variables

These settings express the number of seconds until the timeout occurs. Variables are in milliseconds. Conversion: millisecond = 1; second = 1000; hour = 3600000; day = 86400000; month = 25992000000; no_expire = -1.

Invoice Matching locks records at the application level to prevent multiple users from manipulating the same data. The following settings dictate how long locks on these records can be maintained before timing out. Use any mathematical expression with the time units listed above. For example, 1 * hour.

Table 2–7 Lookout Timeout Variables

Parameter	Description	Comments
business_roles_lock_ timeout	Amount of time until the user's control of (or lock on) the business_roles table ends (times out).	Default is 1 * hour.
reason_codes_lock_timeout	Amount of time until the user's control of (or lock on) the reason_codes table ends (times out).	Default is 1 * hour.
doc_group_list_lock_ timeout	Amount of time until the user's control of (or lock on) the doc_group_list table ends (times out).	Default is 12 * hour.
doc_head_lock_timeout	Amount of time until the user's control of (or lock on) the doc_head_lock table ends (times out).	Default is no_expire.
edi_reject_doc_lock_ timeout	Amount of time until the user's control of (or lock on) the edi_reject_doc table ends (times out).	Default is 1 * hour.
supplier_options_lock_ timeout	Amount of time until the user's control of (or lock on) the supplier_options table ends (times out).	Default is 1 * hour.

Table 2–7 (Cont.) Lookout Timeout Variables

Parameter	Description	Comments
system_options_lock_ timeout	Amount of time until the user's control of (or lock on) the system_options table ends (times out).	Default is 1 * hour.
tolerance_dept_lock_ timeout	Amount of time until the user's control of (or lock on) the tolerance_dept table ends (times out).	Default is 1 * hour.
tolerance_supp_lock_ timeout	Amount of time until the user's control of (or lock on) the tolerance_supp table ends (times out).	Default is 1 * hour.
tolerance_supp_trait_lock_ timeout	Amount of time until the user's control of (or lock on) the tolerance_supp_trait table ends (times out).	Default is 1 * hour.
tolerance_system_lock_ timeout	Amount of time until the user's control of (or lock on) the tolerance_system table ends (times out).	Default is 1 * hour.
receipt_lock_timeout	Amount of time until the user's control of (or lock on) the receipt_lock table ends (times out).	Default is 1 * hour.
parent_invoice_lock_ timeout	Amount of time until the user's control of (or lock on) the parent_invoice table ends (times out).	Default is 1 * hour.

Credit Note AutoMatch Batch Multi-threading Options

Thread pool sizing guidelines are as follows:

- Nthreads = optimal number of threads
- Ncpu = number of available CPUs
- Ucpu = target CPU utilization, 0 <= Ucpu <=1
- W/C = ratio of wait time to compute time
- Nthreads = Ncpu * Ucpu * (1 + ((W/C))

Table 2–8 Credit Note AutoMatch Batch Multi-threading Options

Parameter	Description	Comments
thread.creditnoteautomatch batch.multithreaded	Indicates whether credit note automatch batch processing is in single-threaded mode (False) or multi-threaded mode (True)	Valid values are: true - multi-threaded mode false - single-threaded mode

Table 2–8 (Cont.) Credit Note AutoMatch Batch Multi-threading Options

Parameter	Description	Comments
thread.creditnoteautomatch batch.consumerThreadKeep Alive	The amount of time (in milliseconds) that the batch will keep threads alive after they have completed processing when the current number of runnable threads exceeds the minimum pool size.	Default is 60000.
thread.creditnoteautomatch batch.consumerThread PoolMin	Minimum thread pool size.	Default is 10.
thread.creditnoteautomatch batch.consumerThread PoolMax	Maximum thread pool size.	Default is 20.

Credit Note AutoMatch Workspace Cleanup Setting

Table 2–9 Credit Note AutoMatch Workspace Cleanup Setting

Parameter	Description	Comments
creditnoteautomatchbatch. workspace.cleanup	This setting determines whether IM_ MATCH_* tables are purged after the Credit Note AutoMatch Batch runs. If not purged, data from the previous matching batch process will remain in the workspace tables.	Valid values are: true - After the Credit Note AutoMatch Batch runs, data is purged from IM-MATCH_* tables (except for IM_ MATCH_*_HIST). false - After the Credit Note AutoMatch Batch runs, data remains in the workspace tables.

Invoice AutoMatch Specific Properties

Table 2–10 Invoice AutoMatch Specific Properties

Parameter	Description	Comments
thread.invoiceautomatch batch.threadBy	Indicates the criteria on which invoice automatch threading is based. This setting is depends on the hardware profile and the volume of the thread by groups in implementation (such as areas, locations, and chains). Value should be determined through testing. (String)	Valid values are: NoThread ThreadByLocation ThreadByDistrict (default) ThreadByRegion ThreadByArea ThreadByChain
invoiceautomatchbatch. process.locks	Indicates whether the automatch batch process should exclude locked documents.	Valid values are: true - Locked documents are excluded from matching. false - Locked documents are included in matching.

Generic Threading Options (EdiUpload and AutoMatch)

These threadings settings are expressed in milliseconds. They are utilized by the EdiUpload process and the Invoice AutoMatch batch process.

Table 2–11 Generic Threading Options (EdiUpload and AutoMatch)

Parameter	Description	Comments
thread.backgroundThread Timeout	The amount of time the log-writing thread polls the empty work queue before shutting down. Used only by EdiUpload for rejection files.	Default is 1800000.
thread.consumer ThreadTimeout	The amount of time the consumer pool threads. Used for executing the transactions for both EdiUpload and AutoMatch.	Default is 60000.
thread.consumer ThreadKeepAlive	thread.consumer ThreadKeepAliveThe amount of time (in milliseconds) that the batch will keep threads alive after they have completed processing when the current number of runnable threads exceeds the minimum pool size.	Default is 60000.
thread.consumer ThreadPoolMin	Minimum thread pool size.	Default is 10.
thread.consumer ThreadPoolMax	Maximum thread pool size.	Default is 100.

Translation Caching Timeouts

Invoice Matching caches (or stores) translated descriptions for item names and supplier names, for example.

Table 2–12 Translation Caching Timeouts

Parameter	Description	Comments
translation.caches_refresh_ interval_in_seconds	The number of seconds that elapse before the translation cache is refreshed.	Default is 43200.
translation.locations_desc_ cache_size	The number of entries within the locations' description cache that the system is allowed to use for processing of translated information.	Default is 100000.
translation.suppliers_desc_ cache_size	The number of entries within the suppliers' description cache that the system is allowed to use for processing of translated information.	Default is 100000.
translation.items_desc_ cache_size	The number of entries within the items' description cache that the system is allowed to use for processing of translated information.	Default is 100000.

Logging Configuration Settings

These settings are used only for installation. After installation, they can be changed by manually altering the logging configuration in log4j.properties.

Table 2–13 Logging Configuration Settings

Parameter	Description	Comments
log.online.file	Used to define the path for generating log files.	
log.batch.file	Establishes the name and directory of the batch log file.	
log.batch.error.file	Establishes the name and directory of the batch error files. All errors and routine processing messages for a given program on a given day go into this error file.	
log.level	Indicates the lowest level at which messages should be logged. For example, if value=4, all errors labeled warn, error and fatal are logged.	Valid values are: 2 - fatal 3 - error 4 - warn 5 - validation 6 - info 7 - debug 8 - performance 999 - unknown

Authentication Settings

These settings pertain to user security privileges.

Table 2–14 Authentication Settings

Parameter	Comments
authentication_source	Valid values are: LDAP
IConnectionSettingsDAO	Default is: com.retek.reim.merch.utils.PropertyFileLdap SettingsDao
ISecurityDao	Default is: com.retek.reim.merch.utils.ReIM LdapSecurityDao
ISecurityRelationshipDAO	Default is: com.retek.reim.merch.utils.LDAP SecurityRelationshipDAO
security.ssl_mode	Default is 2.
security.port_non_ssl	Default is 8080.
security.port_ssl	Default is 8443.
security.enable.securejdbc	Indicates whether to enable secure JDBC (requires additional configuration) Default is: true
security.ssl.keyStoreType	Type of the provided key store.
security.ssl.keyStore	Name of key store that contains private keys and certificates.
security.ssl.keyStoreAlias	Alias for credentials to access key store.

Table 2–14 (Cont.) Authentication Settings

Parameter	Comments
security.ssl.trustStoreType	Type of the provided trust store.
security.ssl.trustStore	Name of trust store that contains trusted certificates.
security.ssl.trustStoreAlias	Alias for credentials to access trust store.
sso_url	N/A
sso_conf	N/A
sso_util	N/A
sso_factory_initial	N/A

Online Help

These settings pertain to the online help URL.

Table 2–15 Online Help

Parameter	Description	Comments
online.help.url	Represents the URL for online help.	

Security Settings

These settings pertain to credential store management (CSM).

Table 2–16 Security Settings

Parameter	Description	Comments
csm.wallet.path	Specifies the path on which the wallet resides.	Any path on a filesystem . For example, J:/wallet-store.
csm.wallet.partition.name	Name of the partition inside the wallet, where user name/passwords are mapped against an alias name.	For example, reim13.

system.properties File

This file includes system options settings that cannot be accessed through the graphical user interface (GUI), because they cannot be changed once ReIM has been implemented.

Dynamic / Non-Dynamic GL Options

The parameters in this section of the file determine whether the retailer's segments for the IM_GL_OPTIONS table are dynamic or non-dynamic. Rather than being hard-coded, dynamic segments are populated by company/location or department/class numbers from the invoice. This reduces the amount of maintenance necessary to support posting to the retailer's financial solution.

If the retailer's segments are non-dynamic, all settings are N.

If the retailer's segments are dynamic, note the following:

- The system allows a maximum of four dynamic segments.
- Those GL options that are dynamic can be set up to represent the following
  - Company

- Location
- Department
- Class

Note: Company and location are always paired together, and department and class are always paired together.

The table below illustrates how dynamic GL options are set up to correspond with the retailer's hierarchy parameters. Where mapping occurs, the GL option assumes the value of the corresponding parameter, as determined by the invoice.

Table 2–17 GL Options and Hierarchy Parameters

GL Options	Business Concept Mapping for Dynamic Segments
system.gl_option_dynamic_1=Y	system.gl_option_dynamic_mapping_1=COMPANY
system.gl_option_dynamic_2=Y	system.gl_option_dynamic_mapping_2=LOCATION
system.gl_option_dynamic_3=N	system.gl_option_dynamic_mapping_3=
system.gl_option_dynamic_4=Y	system.gl_option_dynamic_mapping_4=DEPARTMENT
system.gl_option_dynamic_5=Y	system.gl_option_dynamic_mapping_5=CLASS
system.gl_option_dynamic_6=N	system.gl_option_dynamic_mapping_6=
system.gl_option_dynamic_7=N	system.gl_option_dynamic_mapping_7=
system.gl_option_dynamic_8=N	system.gl_option_dynamic_mapping_8=
system.gl_option_dynamic_9=N	system.gl_option_dynamic_mapping_9=
system.gl_option_dynamic_10=N	system.gl_option_dynamic_mapping_10=
system.gl_option_dynamic_11=N	system.gl_option_dynamic_mapping_11=
system.gl_option_dynamic_12=N	system.gl_option_dynamic_mapping_12=
system.gl_option_dynamic_13=N	system.gl_option_dynamic_mapping_13=
system.gl_option_dynamic_14=N	system.gl_option_dynamic_mapping_14=
	system.gl_option_dynamic_mapping_15=
	system.gl_option_dynamic_mapping_16=
	system.gl_option_dynamic_mapping_17=
	system.gl_option_dynamic_mapping_18=
	system.gl_option_dynamic_mapping_19=

Table 2–17 (Cont.) GL Options and Hierarchy Parameters

GL Options	Business Concept Mapping for Dynamic Segments
	system.gl_option_dynamic_mapping_20=

Mapping of Document Types to Action Codes

The table below provides a list of default action codes, based on document type.

Table 2–18 Mapping of Document Types to Action Codes

Parameter	Description	Action Code Default
CRDNT	Credit Note	
CRDNRC	Credit Note Request Price	CBC
CRDNRQ	Credit Note Request Quantity	CBQ
CRDNRT	Credit Note Request TAX Valid Values: CNRTI, if IM_SYSTEM_OPTIONS.TAX_ DOCUMENT_CREATION_ LVL = ITEM CNRTF, if IM_SYSTEM_OPTIONS.TAX_ DOCUMENT_CREATION_ LVL = FULL INVOICE	CNRTI
CRDMEC	Credit Memo Price	CMC
CRDMEQ	Credit Memo Quantity	CMQ
DEBMEC	Debit Memo Price	CBC
DEBMEQ	Debit Memo Quantity	CBQ
DEBMET	Debit Memo Tax Valid Values: DMTI, if IM_SYSTEM_OPTIONS.TAX_ DOCUMENT_CREATION_ LVL = ITEM DMTF, if IM_SYSTEM_OPTIONS.TAX_ DOCUMENT_CREATION_ LVL = FULL INVOICE	DMTI

Labeling Child Invoices

When a parent invoice enters the system, the system can split the invoice into its child invoices. (A parent invoice can contain many locations; a child invoice contains only one.) The retailer determines a string, which the system uses to label a child invoice. This string contains the parent invoice ID plus the system.child_invoice_indicator value plus the location number to which the child invoice is associated.

Table 2–19 Labeling Child Invoices

Parameter	Description	Default
system.child_invoice_indicator	Used in conjunction with the parent invoice ID and location number to label a child invoice.	LOC

Setting the Audit Period

The parameter determines how many days the system retains audit trail data before it is purged.

Table 2–20 Setting the Audit Period

Parameter	Description	Default
system.purge_tolerance_audit_period	Number of days the system retains audit trail data before it is purged	2

Data Translation Options

These options indicate types of language translation within the system.

Table 2–21 Data Translation Options

Parameter	Description	Default
system.language_translation_active	Determines whether the system will perform language translation.	True
system.single_language_translation	Determines whether the system will perform single-language data translation.	SL
system.multiple_language_translation	Determines whether the system will perform multi-language data translation.	ML
system.item_description_language_option	Determines whether the system will perform single-language multi-language data translation of the Description parameter.	SL
system.location_name_language_option	Determines whether the system will perform single-language multi-language data translation of the Location parameter.	SL
system.supplier_name_language_option	Determines whether the system will perform single-language multi-language data translation of the Name parameter.	SL

Set of Books Option

Table 2–22 Set of Books Option

Parameter	Description	Default
system.default_set_of_books_id	The default label assigned to the retailer's set of books.	1

Duplicate Items Option

Table 2–23 Duplicate Items Option

Parameter	Description	Default
system.duplicate_items_lov	Determines whether the same item from a different supplier can appear more than once in the item list of values (LOV).	Y

Maximum Segments Supported

Table 2–24 Maximum Segments Supported

Parameter	Description	Default
system.maximum.segments	Determines the maximum number of segments supported by the system.	20

Application Name

Table 2–25 Application Name

Parameter	Description	Default
system.application.name	Represents the application name.	REIM

Maximum Line Segments

Table 2–26 Maximum Line Segments

Parameter	Description	Default
system.maximum.line. segments	Determines the number of segments to be displayed in the GL cross reference screen.	10

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.

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 throughout the application. Oracle Retail Invoice Matching ships with a sample log configuration that will log basic messages to a standard file located on the application host machine. Retailers wishing to configure specific Invoice Matching loggers should consult the sample configuration log4j.properties file and the Apache Log4j documentation (http://logging.apache.org/log4j).

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. See also "Java Currency Formatting" in Chapter 6, "Technical Design."

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

Language Configuration

The reim.properties file points the application to the location of the user's properties file based on the locale specified for the user in LDAP.

The property file ReIMResources must include the translations for all user interface strings. The translated properties files are identified by the ISO language code for each language. For example: ReIMResources_de.properties contains the German language resources. Translated properties files for Brazilian Portuguese and Traditional Chinese are further differentiated by the ISO language + country code. For example: ReIMResources_pt_BR.properties for Brazilian Portuguese and ReIMResources_zh_TW.properties for Traditional Chinese.

The Java compiler and other Java tools can only process files which contain Latin-1 and/or Unicode-encoded (\udddd notation) characters. The JDK native2ascii tool converts files which contain other character encodings into files containing Latin-1 and/or Unicode-encoded characters. The translated properties files are all shipped in the Unicode-encoded (\udddd notation).

Supported Date Formats

The system's date formats support either two or four digit year designations. Date formats support month name abbreviations or month numbers. Date formats support limited sequencing: year-month-day, month-day-year, and day-month-year. Date formats support either '-' (dash) or '/' (back slash) delimiters. Date formats must be specified in the DateParameters.properties file.

Cache Sizes for Translation Service

To enhance the system's performance speed, the system utilizes a cache when performing data translations into another language.

For example, suppose the system has been configured to offer French translations. When a French user encounters a location name, the system retrieves the translated location name from the database and then stores it in a cache. If the system needs to retrieve the same translated location name at a later time (for another user, for example), the system would retrieve it from the cache rather than from the database. This reim.properties value represents the number of entries within the cache that the system is allowed to use for such processing.

For example:

```
translation.items_desc_cache_size=100000
```

See Electronic Data Interchange (EDI) Tables and Files in Chapter 5, "Integration."

ReIMResources.properties

This file contains a key value pair for every label 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).

3

Technical Architecture

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

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

Overview

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

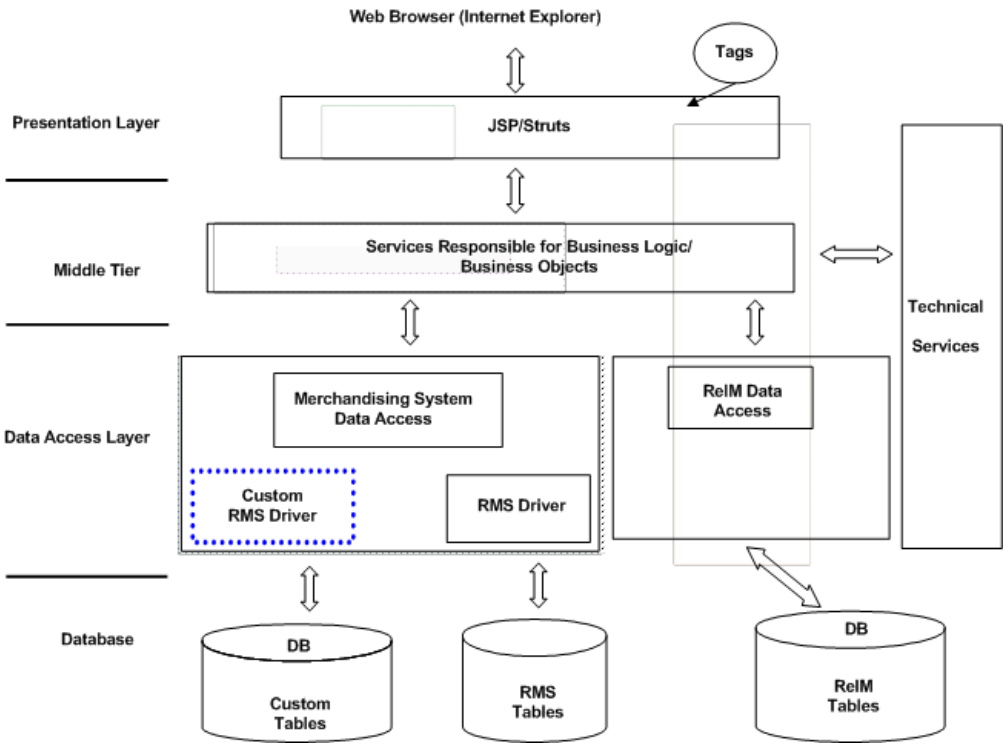
The application is divided into a presentation layer, a middle tier consisting of services and business objects, and a database access/driver layer. Technical services offers the application frameworks for error logging, internationalization, transaction management, application security, and so on.

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

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

The Layering Model

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



Presentation Layer

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

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

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

The presentation layer interacts only with the middle tier services.

Middle Tier

The middle tier comprises the service layer responsible for business logic and Busines Objects.

Service Layer Responsible for Business Logic

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

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

Business Objects

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

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

Data Access Layer (DAL)

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

The ReIM DAL consists of two very distinct portions: a DAL to ReIM owned tables and an interface DAL to merchandising system tables. The two distinct types of Java code are described below.

Database Layer

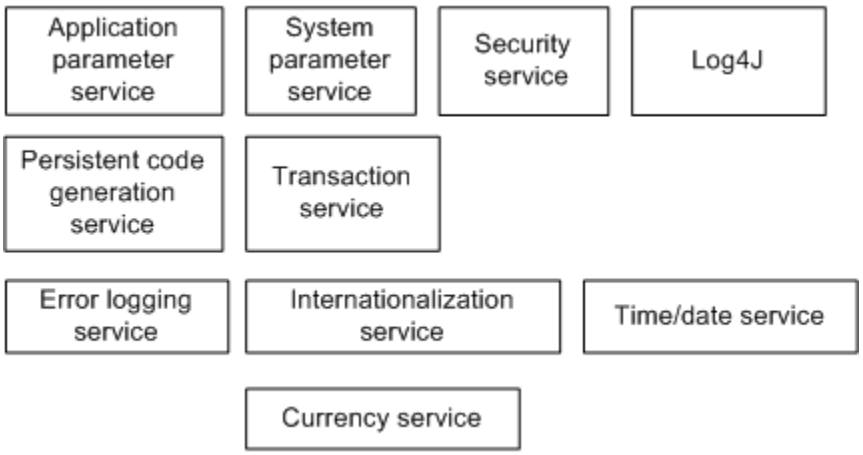
The database layer is the application's storage platform, containing the physical data (user and system) used throughout the application. This layer is only intended to deal with the storage and retrieval of information and is not involved in the manipulation of the data.

Technical Services

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

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

A brief description of each technical service follows the diagram.



Application Parameter Service

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

System Parameter Service

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

Transaction Service

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

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

Error Logging Service

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

The system's coding pattern ensures that the error messages, no matter where they originate, remain detailed in their presentation to the operator.

Log4J

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

Internationalization Service

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

Currency Service

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

Time/Date Service

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

Security Service

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

Third Party Libraries

ReIM base development uses the following third party libraries:

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

ReIM-Related Java Terms and Standards

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

The Java 2 Enterprise Edition (J2EE)

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

Java Database Connection (JDBC)

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

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

Java Development Kit (JDK)

Standard Java development tools from Sun Microsystems.

Java Server Pages (JSP)

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

Java Servlet

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

LOG4J

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

Naming Conventions in Java

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

Struts

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

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

4

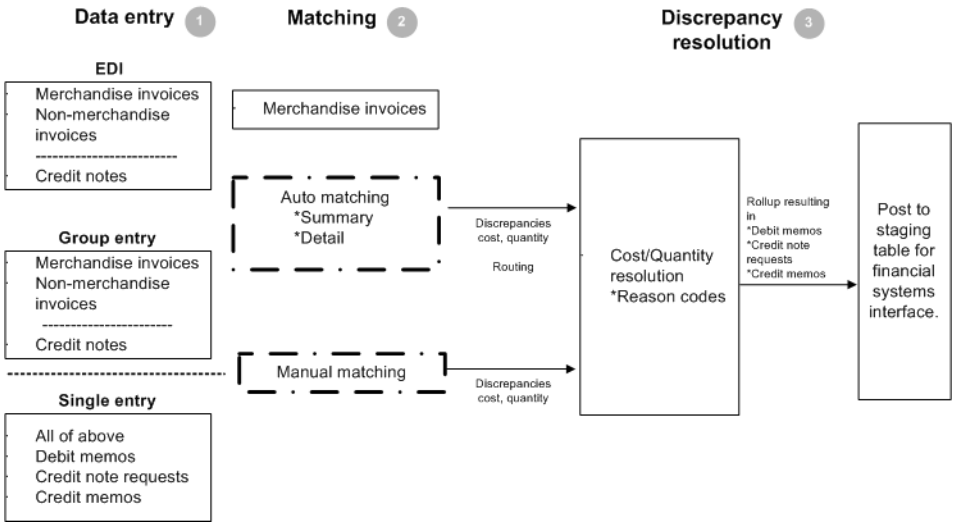
Functional Design

This chapter provides a diagram and description of the invoice matching process flow. It also describes the auto-match process through a series of detailed examples. The various levels of auto-matching are explained, including one-to-one invoice matching and line-level matching. The chapter concludes with a discussion of best terms calculations.

Invoice and Credit Note Matching Process Flow

This section provides a high-level explanation of the process flow in ReIM for each of the following areas:

- Data entry
- Matching
- Discrepancy resolution



Note: Documents drop out of the flow when they require no further processing. For example, if an invoice is matched in Step 2, Matching, the document would not continue to Step 3, Discrepancy Resolution. The document would be posted directly to the financial (AP/GL) staging table after Step 2.

1. Data Entry

   There are three ways in which invoices and other documents enter the ReIM system:

   - **Electronic Data Interchange (EDI)**

     Invoices and credit notes uploaded as part of a batch are assigned a common control number, which is retained on the invoice table as a reference. The control number is assigned by the sender of the EDI file. It is displayed on the Invoice Maintenance screen and may be used for client reporting purposes.

     As necessary, the EDI load process allows for the uploading of supplier's vendor product number (VPN) when neither the document number nor the UPC has been provided. The VPN and the supplier number, then, are used to look up the Oracle Retail item number. ReIM assumes the VPN is related to the supplier associated with the document. Note that the VPN number is not stored in ReIM; it is used to find the Oracle Retail item number which is then retained and used for processing within ReIM.

     Allowing VPN to be used to find the Oracle Retail item number is optional.

     EDI allows ReIM to upload the following documents:

     – **Merchandise Invoices**

       The bills for goods or services received from a supplier or partner. Merchandise invoices may have both of the following:

       Merchandise Costs: Costs that are associated with items on documents. Any other costs on an invoice are non-merchandise costs. The sum of the merchandise costs and non-merchandise costs is the total document cost.

       Non-Merchandise Costs: Costs that are indirectly associated with invoice items, such as freight or handling charges.

     – **Non Merchandise Invoices**

       Bills for non-merchandise costs only (a snow plowing service, for example). Non-merchandise invoices cannot contain items. Either suppliers or partners can create non-merchandise invoices.

     – **Credit Notes**

       Documents received from the supplier, often issued in response to a credit note request from the retailer, which results in a reduction of the retailer's balance owing to a supplier. A credit note request may be raised in lieu of a deduction from invoice (that is, a debit memo) resulting from invoice over-charges, RTVs, rebate bill backs, and so on. Credit notes follow a functional process flow separate from the invoice flow, where credit notes are matched against credit note requests.

   - **Group Entry**

     Group entry facilitates summarized, on-line entry of paper documents. The group entry process accommodates the same types of documents as supported through the EDI process.

     Invoices are entered as part of a batch and assigned a group number, which is retained on the invoice table as a reference. This group number is displayed on the Invoice Maintenance screen and may be used for reporting purposes.

Because group entry is intended to quickly get invoices into ReIM, entry of item details is not required. Adding item details for an invoice can be done later through the Invoice Maintenance screen.

- **Single Entry**

  Single entry is designed as an exception-handling tool made for invoices and documents not entered (for whatever reason) within a group.

  Note: Merchandise invoices entered by way of single entry also are assigned a group/transaction number. However, since each document will be assigned its own group number, some retailers may not want to generate so many additional group IDs. Retailers that require a group/transaction number for tracking purposes may want to restrict access to the single invoice entry screen. Single entry may be controlled for a user group by setting the Invoice Entry option on the User Group Details screen to Modify only. This allows users to change an existing invoice but prevents them from creating a single-entry invoice. In turn, this forces all manual entry to be done as group entry.

  Single entry accommodates the same types of documents supported in the EDI and group entry processes, as well as the following items (if not created automatically through other processes):

  – **Debit Memo**

    A document created to support a deduction from the invoice being paid. Deductions may result from a price or quantity discrepancy. A debit memo also refers supplier billing for rebates, RTVs, and so on. Debit memos also can be created as 'stand-alone' documents (that is, created on-line, but not supported by any processes in ReIM or the merchandising system).

  – **Credit Note Request (CNR)**

    A document sent from the retailer to the supplier, requesting a credit note for an over-invoiced amount (discrepancy) or in support of various billing activities (for example, rebates, RTVs). If a credit note request is not satisfied by the supplier in a timely manner, ReIM provides the ability to convert it into a debit memo (and include the number of the invoice to which it is assigned). Credit note requests also may be created as stand-alone documents.

  – **Credit Memos**

    A document created to refund a supplier for an under-invoiced or over-billed amount (for example, for rebates not meeting threshold performance levels) amount. Credit memos also may be created as stand-alone documents.

  Note: If the credit memo is the result of a reversed debit memo, the ID number of the invoice to which the debit memo is associated should be assigned to the credit memo, particularly if the invoice is being held for payment. Assigning the ID number in this manner ensures related documents are released to accounts payable at the same time.

  ---

2. **Matching**

Note: Credit notes must be matched on-line against credit note requests. Credit note matching is not supported by the automatic matching process.

- **Auto-Matching**

  Merchandise invoices are grouped by common PO/location; ReIM requires these attributes in all merchandise invoices. ReIM accesses the merchandising system to determine what shipments (receipts) were created for the PO/location. The auto-matching process attempts to support invoice cost and quantities against receipt quantities at PO cost within user defined tolerances.

  If the auto-matching process identifies cost or quantity differences outside of the pre-established tolerance range, the system creates corresponding discrepancies (cost or quantity). Otherwise, matched invoices are posted to the financial staging table.

  For header-level-only invoices, TAX validation is performed as a final validation step, after cost and quantity matching has been performed.

  For more functional information about summary and detail-level auto-matching, see "The Auto-Match Process" in this chapter.

- **On-line Matching**

  – **Invoices**

    The on-line matching dialog provides users with the ability to match invoices with even greater flexibility than the auto-match process. Invoices are initially grouped by their PO/location, but the groups can be modified beyond the common PO/location relationship based on available (that is, 'unmatched') invoices and receipts, to support matches.

    On-line matching either matches a document, which is posted to the financial staging table, or supports creation and resolution of a cost and/or quantity discrepancy.

  – **Credit Notes and CNRs**

    Typically, invoices for which CNRs are generated are sent to accounts payable even if matching credit notes have not yet been received. The retailer, then, is issued an invoice that actually is higher than it should be and will have to wait until credit notes are processed before receiving credit for the overcharge. The supplier, in turn, may be overpaid. To avoid this inefficiency, ReIM allows invoices with unmatched CNRs to be held (not paid) until all corresponding credit notes are received-at which time the invoice automatically is sent to accounts payable. Depending on user group security, the user can manually control when the invoice is released to accounts payable-even before all credit notes are received.

    When a credit note request is matched to a credit note through online matching, the ID number of the invoice to which they are associated is assigned to the credit note. In this way, the invoice and all related documents may be released to accounts payable at the same time.

    When matching CNRs to credit notes on a held invoice, the original invoice should be checked for other open discrepancies. If none exist, the Hold Invoice indicator on the Supplier Options screen should be turned

off so that the invoice and all related documents can be released to the financial system.

3. **Discrepancy Resolution**

   Users assign pre-defined reason codes against cost and quantity discrepancies to support resolutions. The reason codes direct the system to take a specific action.

   Cost and quantity discrepancies are routed to on-line lists by user group. (Pre-established user groups and routing rules determine which discrepancies populate which user group list.) For example, in many companies the merchant/buyer is responsible for verification of invoice cost against the PO. To support this functionality, a user group of buyers by department or class might be a logical association to assign to an on-line Cost Discrepancy Review List. (Each user group would see only discrepancies assigned to it). Each user group is empowered to resolve discrepancies according to their authorization. Similarly, it may be logical to assign users groups to Quantity Discrepancy Review Lists based on receiving location.

   ReIM does not require the resolution of discrepancies through the routing process; the application will support a more centralized business process for resolving discrepancies using only the on-line matching dialog.

   Once all discrepancies are resolved for the document, it is posted to the financial staging table along with any corresponding debit memos, and so on, for posting to the retailer's accounts payable solution.

   Documents supporting discrepancy resolution (such as debit memos, credit note requests, and credit memos) are available for EDI download to the supplier. (Or the retailer may develop reporting from these values stored in the ReIM tables). These document records (except credit note requests) also are posted to the financial staging table.

   If there is a discrepancy between a credit note and a credit note request, a new credit note should be created. Further, CNRs created inadvertently can be voided and fully reversed to expedite resolution. (It is assumed that if all CNRs related to a held invoice are voided, that invoice is released for payment.)

The Auto-Match Process

The Auto-Match process includes invoices and credit notes.

- **Invoices**

  Invoices in ready for match, unresolved, and multi-unresolved status are retrieved from the database to be processed through the auto-match algorithm. These invoices are grouped with receipts based upon PO/location.

  If no receipts exist for the PO/location, invoices process through the cost pre-matching algorithm.

  If receipts do exist, the system attempts to match all invoices and receipts for the common PO/location (referred to as'group matching), within summary-level tolerances.

  If group matching fails, the system attempts to match each invoice to a single receipt in the one-to-one matching algorithm. If all invoices are matched in this fashion, then the next PO/location is processed.

If all of the invoices cannot be matched and a multi-unresolved scenario results, the matched invoices remain matched and the non-matched invoices are given a multi-unresolved status. No further processing occurs for this PO/location.

If an unmatched invoice is eligible for line level matching, an attempt is made to match each line on the invoice to an unmatched receipt line.

- **Credit Notes**

  Credit notes can be matched on-line against credit note requests or by using the credit note auto-matching process as described in the "Credit Note Auto-Matching" section below.

TAX on Header Level Only Invoices

The auto-matching process determines whether the TAX values on header level-only invoices are correct. The system only processes invoices that do not have any unresolved TAX discrepancies.

The invoice status determines whether an invoice can be processed by the Auto-match batch process (AutoMatchService). Only invoices in ready-for-match status are processed. Those with a status of TAX discrepancy are not processed by the batch. See Chapter 9, "Batch Processes," for information.

Invoices created without details are not able to have their TAX information validated at invoice creation. All header level-only invoices are created with a status of ready-for-match. These invoices must have a TAX validation executed as part of the invoice matching process. This validation determines whether a header level-only invoice that was matched to a receipt should continue in the matching and posting process or whether it should be marked as having a TAX discrepancy and removed from the matching process.

Cost Pre-Matching

Cost pre-matching occurs only for PO locations that meet the following conditions:

- Invoices that have never been processed by auto-match exist.
- No receipts exist.

Each invoice line unit cost is compared with the PO item location's unit cost. If the unit costs match within tolerance, the invoice and lines are processed again by auto-match once receipts come in for the PO location.

If there is a discrepancy, then the invoice is processed again once receipts arrive. However, the lines that contain a discrepancy are immediately routed for cost resolution. Once invoices are run through the cost pre-matching algorithm, they are not re-run when the next auto-match run occurs if there are still no receipts.

Scenarios can arise where no receipt lines exist, and no order line corresponds to an invoice line. The assumption is that validation occurs in the EDI upload process and in the manual invoice entry screens prevent these invoices from entering the system. Therefore, auto-match ignores this situation.

PO/Location Summary Group Matching

PO/location summary group matching processes the following:

- Invoices that have never been processed before by auto-match.

- Invoices that have been processed previously by auto-match but remain unresolved.
- Invoices that have been processed previously by auto-match but that have been identified as multi-unresolved.

First, the system attempts to match the total extended cost of the invoices with the total extended cost of the receipts. Extended cost is defined as the unit cost for an item multiplied by the quantity received or the quantity invoiced. For this comparison, all extended costs are summed for the group of invoices and receipts and compared. The total extended cost for each invoice is taken from the invoice header. The process, however, calculates the receipts' total extended cost.

Quantity matching is also sometimes required. Whether quantity matching is performed is determined by a supplier option. Quantity matching compares the total quantity invoiced for the PO location with the total quantity received for the PO location. As in cost matching, the total quantity invoiced for each invoice is taken from the invoice header. For receipts, the process calculates this sum.

For invoices with quantities representing weight rather than number of eaches, total quantity displayed in the invoice header is represented as the sum of item quantities in abstract UOM.

Auto-match processing first attempts to match the total extended costs, and optionally the total quantities, exactly. If the costs and quantities do not match exactly, then the system attempts to match them within tolerance. If a match is achieved, all of the invoices, receipts, and their lines for the PO location are assumed to be matched. If a match is not achieved, all invoices and receipts for the PO location are unresolved. These invoices and receipts are processed further with one-to-one invoice matching.

Auto-match accounts for the actions taken by cost reviewers that fully resolve a cost discrepancy when attempting to match at the summary level. If a match is achieved at the summary level, auto-match deletes any outstanding unresolved cost discrepancies and any partially resolved cost discrepancies along with their partial resolutions for the PO location from the system.

Example 1

The following example illustrates a successful match:

Invoices for a PO/Location	Total Extended Cost	Total Quantity
Invoice 1	$50,000	1,000
Invoice 2	$150,000	5,000
Totals:	$200,000	6,000

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Receipt 1	$50,000	2,000
Receipt 2	$50,000	2,000
Receipt 3	$100,000	2,000
Totals:	$200,000	6,000

In the example, the total extended costs and the total quantities match for the PO location. Therefore, all invoices and receipts will be set to matched status.

Example 2

The following example illustrates a successful match, but where quantity matching is not required by the supplier.

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Invoice 1	$50,000	2,000
Invoice 2	$150,000	5,000
Totals	$200,000	7,000

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Receipt 1	$50,000	2,000
Receipt 2	$50,000	2,000
Receipt 3	$100,000	2,000
Totals	$200,000	6,000

In the example, only the total extended costs match. However, quantity matching is not required for this supplier. Therefore, these invoices and receipts are considered matched by the auto-matching algorithm.

Example 3

The following example illustrates an unsuccessful match, where quantity matching is required by the supplier.

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Invoice 1	$50,000	1,000
Invoice 2	$150,000	5,500
Totals:	$200,000	6,500

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Receipt 1	$50,000	2,000
Receipt 2	$50,000	2,000
Receipt 3	$100,000	2,000
Totals:	$200,00	6,000

In the example, because quantity matching is required for the supplier, the match is unsuccessful despite the fact that the costs do match. The invoices and receipts will be set to unresolved status, and an attempt will be made to match them at a one-to-one level.

Example 4

The following example illustrates a successful match within tolerance.

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Invoice 1	$50,035	1.000
Invoice 2	$150,100	5,000
Totals:	$200,135	6,000

Receipts for a PO/Location	Total Extended Cost	Total Quantity
Receipt 1	$50,000	2,000
Receipt 2	$50,000	2,000
Receipt 3	$100,000	2,000
Totals:	$200,000	6,000

One-to-One Invoice Matching

One-to-one invoice matching attempts to match each invoice for the PO/location with a single receipt for the PO/location. First, the system attempts a match between the total extended costs. If the extended costs match, the system may, depending upon a supplier option, attempt a match between the total quantities. If there is either an exact match or a match within tolerance, the invoice and receipt along with their lines are considered to be matched. If no match can be found for the invoice, it is left unresolved.

One-to-one matching may result in a multi-unresolved scenario. If any invoices within the PO/location can be successfully matched with one and only one receipt and that receipt can be matched to only one invoice for the PO location, then those invoices and receipts are considered to be matched. If no unmatched invoices remain, then processing stops for the PO/location and all invoices are considered matched. Only when one unmatched invoice exists for the PO/location can line level matching occur. If more than one invoice remains after one-to-one matching, then all remaining unmatched invoices and receipts are considered to be multi-unresolved.

One-to-one matching is driven by invoices. Therefore, if there are unmatched receipts remaining but no unmatched invoices for the PO/location, no further processing occurs. The receipts remain unresolved but no discrepancies are generated.

Example 1

The following example illustrates how one invoice matches with one and only one receipt. One invoice and two receipts are unresolved.

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Invoice 1	$50,000	5,000	Matched
Invoice 2	$100,000	10,000	Unresolved

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 1	$50,000	5,000	Matched
Receipt 2	$25,000	2,500	Unresolved

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 3	$35,000	2,500	Unresolved

In the example, Invoice 1 matches with Receipt 1. However, the remaining invoice and receipts do not match one-to-one. Because there are two unmatched receipts remaining and only one unmatched invoice, the remaining unmatched invoice and receipts are considered to be unresolved. If they are eligible for detail matching, they are sent to the detail-matching algorithm.

Example 2

The following example illustrates a multi-unresolved match, with no successful matches.

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Invoice 1	$50,000	5,000	Multi-unresolved
Invoice 2	$25,000	2,500	Multi-unresolved
Invoice 2	$35,000	3,000	Multi-unresolved

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 1	$40,00	4,000	Multi-unresolved
Receipt 2	$25,000	2,500	Multi-unresolved
Receipt 3	$25,000	2,500	Multi-unresolved
Receipt 4	$10,000	1,000	Multi-unresolved

In the example, Invoice 2 can be successfully matched to both Receipt 2 and Receipt 3. Therefore, no match can be obtained for Invoice 2. All invoices and receipts are set to multi-unresolved status.

Example 3

The following example illustrates another multi-unresolved match, with no successful matches.

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Invoice 1	$40,000	4,000	Multi-unresolved
Invoice 2	$25,000	2,500	Multi-unresolved
Invoice 3	$25,000	2,500	Multi-unresolved
Invoice 4	$10,000	1,000	Multi-unresolved

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 1	$50,000	5,000	Multi-unresolved
Receipt 2	$25,000	2,500	Multi-unresolved

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 3	$35,000	3,500	Multi-unresolved

In the example, Receipt 2 can be successfully matched to both Invoice 2 and Invoice 3. All invoices and receipts are set to multi-unresolved status.

Example 4

The following example illustrates a multi-unresolved match, but one with successful matches.

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Invoice 1	$50,000	5,000	Multi-unresolved
Invoice 2	$25,000	2,500	Multi-unresolved
Invoice 3	$35,000	3,500	Matched

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 1	$40,000	4,000	Multi-unresolved
Receipt 2	$25,000	2,500	Multi-unresolved
Receipt 3	$25,000	2,500	Multi-unresolved
Receipt 4	$35,000	3,000	Matched

In the example, Invoice 2 can be successfully matched with both Receipt 2 and Receipt 3. Invoice 3, however, can be successfully matched only with Receipt 4. Therefore, Invoice 3 and Receipt 4 are set to matched status. All other invoices and receipts for the PO location are set to multi-unresolved status.

Example 5

The following example illustrates a scenario in which all invoices match, but there are remaining unresolved receipts.

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Invoice 1	$50,000	5,000	Matched
Invoice 2	$25,000	2,500	Matched
Invoice 3	$35,000	3,000	Matched

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 1	$50,000	5,000	Matched
Receipt 2	$25,000	2,500	Matched
Receipt 3	$15,000	2,500	Unresolved
Receipt 4	$35,000	3,000	Matched

Invoices for a PO/Location	Total Extended Cost	Total Quantity	Status Post Matching
Receipt 5	$75,000	10,000	Unresolved

In the example, all three invoices can be successfully matched to one and only one receipt. However, two unmatched receipts remain. The invoices are still considered matched, and the receipts remain unresolved.

Elibigility for Line-Level Matching

In auto-matching, matching can be performed for entire invoices or broken down to the line level. PO location level matching and one-to-one invoice matching are performed for entire invoices and receipts. Line level matching is performed by item.

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

1. Neither the invoice nor receipt can be in multi-unresolved status: If the invoice or receipt is in multi-unresolved status, it is assumed that human intervention is required. No further attempts are made to match the applicable invoice at the line level.
2. Lines must be present on the invoice: Auto-matching assumes that invoices either have all lines in the system or no lines. The system neither validates nor processes partial invoices. If any lines are present, auto-matching assumes that all lines are present.
3. The number of days to routing must be exceeded: The system uses settings and a formula to arrive at its determination of routing days. A supplier option is used to define how long the system should wait before routing discrepancies for invoices for that supplier. However, if the invoice is due sooner than the routing date, then discrepancies may be routed earlier than the route date. A system option determines the maximum number of days before an invoice due date that discrepancies will be routed. The earliest date between the routing date defined by the supplier option and the routing date dictated by the system option is the date on which auto-match routes discrepancies for an invoice.

   Supplier option: routing days = x days

   System option: maximum days before due date = y days

   Supplier driven routing date = invoice date + x days

   System driven routing date = invoice due date - y days

   The date of actual routing is the earlier of the supplier driven routing date and the system-driven routing date.

Line-Level Matching

If only one invoice remains unmatched and zero-to-many receipts are unmatched for the PO location and the invoice is eligible, the system attempts to match each line item on the invoice to receipt line items on the receipts for the same item. If a match is not found, price and/or quantity discrepancies are created and routed. Once line level matching is complete for a PO location, if all lines have been matched, then the entire invoice and all of the receipts are considered matched. Otherwise, they remain unresolved.

When invoice lines are sent through line level matching, all existing unresolved or partially resolved cost discrepancies are deleted along with any partial resolutions. If line level matching produces new discrepancies, they are created and routed, thus ensuring that discrepancies are routed with the latest information available about the invoice and receipt lines.

If no receipt lines correspond to an invoice line, cost matching is attempted for the applicable invoice line using the unit cost from the PO. The system assumes the invoice line exists on the order. If there is a discrepancy, a cost discrepancy is created and routed. In this scenario, a quantity discrepancy is automatically created and routed where the entire invoiced quantity is the discrepant quantity.

For line-level matching, cost and quantity matching are always performed. If cost matching fails, quantity matching is still performed in order to route potential quantity discrepancies that may be discovered. When discrepancies are created, the PO supplier is associated with the discrepancy.

For quantity line level matching, the comparison is made between the quantity invoiced and the sum of the quantities received across the receipts for that item. If a quantity match cannot be obtained, then a quantity discrepancy is generated and routed for the invoice line and the receipt lines for that item.

Example 1

The following example illustrates a scenario in which all lines match, and the invoices and receipts are set to matched status.

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Invoice 1			550	Matched
- Invoice line	Item 1	$5.00	100	Matched
- Invoice line	Item 2	$10.00	200	Matched
- Invoice line	Item 3	$15.00	250	Matched

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Receipt 1			565	Matched
- Receipt line	Item 1	$5.02	105	Matched
- Receipt line	Item 2	$10.10	210	Matched
- Receipt line	Item 3	$15.03	250	Matched

In the example, assume line-level tolerances are set such that all lines match, and the line-level statuses are set to matched accordingly.

Example 2

The following example illustrates a scenario in which some lines match, and the invoices and receipts remain in unresolved status.

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Invoice 1			550	Unresolved
- Invoice line	Item 1	$12.00	100	Unresolved

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
- Invoice line	Item 2	$10.00	200	Matched
- Invoice line	Item 3	$12.00	250	Unresolved

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Receipt 1			550	Unresolved
- Receipt line	Item 1	$5	100	Unresolved
- Receipt line	Item 2	$10	200	Matched
- Receipt line	Item 3	$10	250	Unresolved

In the example, the lines value for Item 2 is matched. However, because Items 1 and 3 do not match within tolerance, the receipt and invoice are unmatched.

Example 3

The following example illustrates a scenario in which some lines match, and the invoices and receipts remain in unresolved status. Note that one invoice line has no corresponding receipt item.

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Invoice 1			550	Unresolved
- Invoice line	Item 1	$12.00	100	Unresolved
- Invoice line	Item 2	$10.00	200	Matched
- Invoice line	Item 3	$12.00	250	Unresolved

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Receipt 1			550	Unresolved
- Receipt line	Item 1	$5.00	100	Unresolved
- Receipt line	Item 2	$10.00	200	Matched

Order Lines for a PO/Location	Item	Unit Cost
- Order line	Item 1	$5.00
- Order line	Item 2	$10.00
- Order line	Item 3	$12.00

In the example, Item 2 matches. A cost discrepancy is created for Item 1. No cost discrepancy is created for Item 3 because its unit cost matches the PO unit cost. A quantity discrepancy is created for Item 3 where the received quantity is zero because the item is not on the receipt.

Example 4

The following example illustrates a scenario in which one invoice line is matched to many receipt lines.

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Invoice 1				Matched
- Invoice line	Item 1	$5.00	100	Matched

Invoice Lines for a PO/Location	Item	Unit Cost	Quantity	Status Post Matching
Receipt 1				Matched
- Receipt line	Item 1	$5.00	70	Matched
Receipt 2				Matched
- Receipt line	Item 1	$5.00	30	Matched

In the example, one invoice line can be matched with two receipt lines.

Recycling and Overall Flow

As soon as invoices arrive, the next auto-matching batch run processes them. If there are no receipts, invoices are sent to cost pre-matching immediately - allowing for early identification of cost discrepancies and correction of PO, if necessary, to improve match rates when receipts arrive.

Once receipts arrive, the invoices and receipts are matched at the PO/location level and at the one-to-one level. If no match exists, these invoices and receipts are recycled through summary level matching until the routing days parameter has passed. If there is a match, then any unresolved or partially resolved cost discrepancies are removed from the system.

For discrepancies that have been fully resolved, the actions taken are reflected in the adjusted total extended cost and adjusted total quantity of the invoices and the receipts. The adjusted cost and quantity values will be available to support summary matching on-line.

After the routing days parameter has passed, an invoice in unmatched status will undergo line-level matching. In this type of scenario, all existing unresolved or partially resolved cost discrepancies are deleted. New cost and quantity discrepancies are created if any exist.

After line level matching is performed for an invoice (through either the auto-match or the on-line matching process), that invoice is never processed by auto-match again.

Partially Matched Receipts

Users may choose to split a receipt item. Splitting a receipt a portion of the receipt quantities to support a match or resolve a discrepancy online. The unmatched portion of the receipt is available to match against future invoices. Partially matched receipts (that is, the unmatched portion) are available to both on-line and auto-match processes to support matches.

Example 1

The following example illustrates summary level matching:

Invoice Lines for a PO/Location	Item	Extended Cost	Quantity	Status Prior to Matching	Status After Matching
Invoice 1		$30,000	500	Unresolved	Matched

Invoice Lines for a PO/Location	Item	Extended Cost	Quantity	Status Prior to Matching	Status After Matching
		$60,000	1,000	Partially Matched	Matched
		$30,000	500	Previously matched	Matched
		$15,000	250	Unresolved	Matched
		$15,000	250	Unresolved	Matched

In the example, a partially matched receipt is used to match an unprocessed invoice. Only the unmatched lines for the receipt are used to determine whether the invoice and receipt match at the summary level.

Example 2

The following example illustrates line level matching:

Invoice Lines for a PO/Location	Item	Extended Cost	Quantity	Status Prior to Matching	Status After Matching
Invoice 1				Unresolved	Unresolved
- Invoice line	2	$15.00	250	Unresolved	Unresolved
- Invoice line	3	$15.00	250	Unresolved	Matched

Invoice Lines for a PO/Location	Item	Extended Cost	Quantity	Status Prior to Matching	Status After Matching
Receipt 1				Partially Matched	Matched
- Receipt line	Item 1	$30.00	500	Previously matched	Matched
- Receipt line	Item 2	$15.00	250	Previously matched	Matched
- Receipt line	Item 3	$15.00	250	Unresolved	Matched

In the example, the invoice remains unresolved and the receipt becomes matched. Even though Item 2 of Invoice 1 matches with Item 2 of Receipt 1, Item 2 of Receipt 1 had already been matched to a different line on a different invoice. Therefore, it is not reused here to make a match. Item 3 of Receipt 1 is unresolved and is therefore available to be matched to Item 3 of Invoice 1.

Matching Tolerances

Matching tolerances are defined for:

- Costs and quantities, for both summary and detail (line-level) matching
- Discrepancies in favor of the retailer and those in favor of the supplier
- Tolerance ranges
- Supplier, department, or system level (default)

Tolerances are set up for total invoice (merchandise) cost to support summary level matching during the auto-match and on-line processes. Summary level quantity matching is optional, per supplier parameter; the tolerance dialog. Detail (line-level) cost matching is performed based on the unit cost of the item. Quantity matching is always done at the line-level, requiring a tolerance provision. Tolerances may be set up as percentages or nominal amounts. A system option provides for definition of the maximum percentage tolerance that can be used.

Tolerances are defined separately for discrepancies in favor of the retailer and those in favor of the supplier. To illustrate, if the invoice cost is $20.00 and the purchase order (receipt) cost is $30.00, the discrepancy of 10 is in favor of the retailer because the invoice cost is less than expected.

Discrete tolerance amounts or percentages may be defined for value and quantity ranges to hone the matching process. Ranges are defined for summary and line-level matching.

In general, when attempting to match invoices with quantities representing weight, the percentage tolerance type at the system level should be used For these invoices, the system would apply the same logic regardless of whether the invoice shows number of pounds or number of packs. So using a percentage rather than an amount would be the more effective choice.

Tolerances may be defined at the supplier level, the department level, or the system-wide level. The matching processes first determine whether a supplier-level tolerance applies to the invoice being matched. If a supplier-level tolerance does not exist, a check will be made for department level. If line-level matching is being performed, then ReIM will use the department for that item to retrieve tolerances. In the summary matching case, an item is selected at random from the corresponding PO, and the department for that item is used to retrieve tolerances. Finally, if supplier- and department- level tolerances do not exist, the system-level tolerance will default.

History and Metrics

ReIM records summary and detail history for matched invoices and receipts. In addition, the system records whether or not each match was exact or not. At the summary level, the group of receipts and invoice numbers is stored. At the detail level, receipt lines matching to a particular invoice line is also stored.

For each auto-match run, the following metrics about the run are stored:

- The run date.
- The number of invoices that matched exactly.
- The number of invoices that matched within tolerance.
- The total number of invoices processed.

Best Terms Calculations

The best terms calculation process compares the terms on the invoice and the terms on the PO, selects the most favorable term (according to each term's ranking), and determines a terms date. Best terms and terms date are subject to supplier-level option. The best terms calculation process is called after the auto-matching and on-line matching processes, and for pre-paid invoices.

After the best terms are calculated and the terms date is determined, the results are written to IM_DOC_HEAD for the invoice.

Terms Ranking Overview

Terms are ranked numerically. Terms with a lower ranking are preferable to terms with a higher ranking. During the best terms calculation, the ranks of the invoice terms and the PO terms are compared, and the terms with the lowest rank are selected as the best terms.

Supplier Options

The following supplier options (IM_SUPPLIER_OPTIONS) affect the best terms calculation:

- Always Use Invoice Terms (USE_INVOICE_TERMS_IND): When this indicator is set to Y, only invoice terms will be used, and there the comparison against PO terms is not performed.
- ROG Date Allowed (ROG_DATE_ALLOWED_IND): When this indicator is set to Y, the supplier allows the receipt of goods (ROG) date to be used to when determining the terms date. This indicator can only be set to Y if the Always Use Invoice Terms indicator is set to N.

Terms Date

The terms date is the later of the invoice date or the receipt of goods (ROG) date. The ROG date replaces invoice date as the terms date when all of the following are true:

- Always Use Invoice Terms (USE_INVOICE_TERMS_IND) on the supplier options table is set to N.
- ROG Date Allowed (ROG_DATE_ALLOWED_IND) on the supplier options table is set to Y.
- The ROG date is later than the invoice date.

Note: If there are multiple receipts for an invoice, the ROG date is the date of the last receipt.

Assumptions and Dependencies

- Best terms calculation applies only to merchandise invoices.
- Merchandise invoices must be in matched status before the best terms calculation is performed.
- When the supplier option Always Use Invoice Terms is set to Y, the invoice terms are always used. The due date is calculated using the invoice date. The PO terms are never considered.
- The payment of invoices prior to or during matching does not update the matching status of the invoice. In these situations, a pre-paid invoice indicator is tripped to ensure the invoice is not paid a second time after matching and to trigger the correct accounting distribution. The best terms process is not re-invoked if the pre-paid indicator is set to Y.

Credit Note Auto-Matching

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.

The following table describes under which circumstances credit notes and credit note requests are eligible to be matched by the CreditNoteAutoMatchBatch process.

Table 4–1

Document Status	Document Hold Status	Holding Supplier	Credit Notes	Credit Note Requests
Approved	Never held	Yes	N/A	Eligible
Approved	Held	Yes	Eligible	N/A
Approved	Released	Yes	Eligible	Eligible
Approved	Never held	No	Ineligible	N/A
Posted	Never held	No	Eligible	N/A

In addition to the requirements listed above, the following criteria must apply for documents to be processed by the CreditNoteAutoMatchBatch:

- Credit notes must never have had a discrepancy created against them.
- Credit notes must never have been previously detail matched.

If the documents are eligible for matching, they are collected into a pool of matchable documents by the batch. The batch process is multi-threaded. It performs matching on eligible documents by first grouping the eligible documents with respect to the supplier. Once grouped with respect to the supplier, the documents are processed for each configurable key. Each document-key set is further processed using the following three matching algorithms.

- Summary matching
- One-to-one matching
- Detail (line level) matching

In summary matching, documents are matched in groups at the summary level by comparing the total extended costs for all the documents in the group. Quantity matching is only attempted if the supplier options indicate it as required. If a match is achieved, the documents are marked with the matched status, and drop out of the matching pool.

If the summary match attempt fails for the group, the batch attempts matching at the one-to-one level. One-to-one matching attempts to match each distinct eligible credit note to a single credit note request. The match is again attempted by comparing the extended cost on the credit note to that of the credit note request, and quantity matching is only attempted depending on the supplier options. If one-to-one matches are found, they are flagged as such and will not be processed by subsequent match attempts.

Line level matching is the last attempted match algorithm. This algorithm attempts to match documents at the item line level. To avoid item matching between unrelated credit notes and credit note requests, this algorithm expects just one unmatched credit note to be remaining in the matchable pool. In case there is more than one credit note still in unmatched status, no match attempt will be made. Line level matching also automatically creates and resolves discrepancies, if the appropriate system options

have been set. Once these discrepancies are created, the algorithm also attempts to resolve the discrepancies by creating resolution actions in the system in accordance with the nature of the discrepancies.

On the next run of the ReasonCodeActionRollupBatch, documents are generated for any resolution action generated by the CreditNoteAutoMatchBatch.

Configurable Keys (Flexible Pool Keys)

Grouping documents into sets using configurable (flexible) pool keys allows for matching in combinations beyond just the PO / Location combination. Note that when we refer to a document set, the set can only contain documents within the same supplier.

These document-key sets are categorized by common attributes which are defined on the document itself (credit notes and credit note requests). These attributes are referred to as Configurable or Flexible Pool Keys. By default, the credit note auto match process aggregates document sets based on the following keys:

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

In case of Credit Note Request ID and Original Invoice ID, two of the four customizable reference fields of the documents are used as place holders for the key values. For instance, the Ref No.3 field is used to store the Credit Note Request ID, and the Ref No. 4 field is used to store the original Invoice ID.

A document can exists in only one document-key set at a time. Note that a document-key set will exist only if it contains both credit notes and credit note requests. Matching will be attempted only for sets not containing both credit notes and credit note requests. This makes it impossible to create, route and resolve discrepancies for credit notes that are yet to be received by the retailer.

Within each document-key set, matches are attempted using three different matching algorithms. If a match is obtained with an algorithm, the matched documents are flagged as such, and processing continues on to the next document-key set. When all configurable three algorithms are finished processing within a document-key sets, processing moves to the next configurable key-set and starts again from the first matching algorithm.

Below is the order for attempting a match in a document-key set when no match is found:

1. Credit Note Request ID (configurable key)
    - Summary Matching (matching algorithm)
    - One to One Matching (matching algorithm)
    - Line -level Matching (matching algorithm)
2. Original Invoice ID (configurable key)
    - Summary Matching (matching algorithm)
    - One to One Matching (matching algorithm)
    - Line -level Matching (matching algorithm)
3. PO / Location (configurable key)
    - Summary Matching (matching algorithm)

- One to One Matching (matching algorithm)
- Line -level Matching (matching algorithm)

Summary Group Matching Algorithm

Summary matching attempts to match the total extended cost of the credit notes with the total extended cost of the credit note requests. Extended cost is defined as the unit cost for an item multiplied by the quantity of the item on the document. The total extended cost for each credit note and credit note request is taken from the document header.

Quantity matching also is sometimes required. Whether quantity matching is performed is determined by supplier options. Quantity matching compares the total quantity on the credit note, with the total quantity on the credit note request. As in cost matching, the total quantity for each credit note and credit note request is taken from the header.

If the costs and quantities do not match exactly, then the system attempts to match them within tolerance. See "Matching Tolerances" for more details. If a match is achieved, all of the credit notes, credit note requests, and their lines for that document-key set are assumed to be matched. If a match is not achieved, all credit notes and credit note requests for that document-key set are left in their original approved status. After summary matching has been completed, the credit notes and requests become eligible to be processed with a different matching algorithm if no match was found at the summary level.

Consider an attempted summary match where two credit notes were received for two credit note requests. Assuming that the invoice and receipt data in the application is as follows:

- Purchase Order=89890
- Location=1000001

Table 4–2 Attempted Summary Match

Document Type	Document ID	Unit Cost	Extended Cost	Quantity
Invoice	INV555	$11.00	$440	40
Receipt	SHP444	$10.00	$200	30

When matched, the invoice and receipt will create a cost discrepancy of $40, and a quantity discrepancy of $100--which will generate a credit note request cost for $40 and a credit note request quantity for $100.

The default Invoice Matching Pool Key configuration will have the following priority and values.

1. Credit Note Request ID
2. Invoice ID
3. Purchase Order/Location

Example 1: Matchable by credit note request ID; quantity matching not required by supplier.

The following example illustrates a successful match using the first Pool Key attribute, Credit Note Request ID.

Table 4–3

Credit Note Request ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNRC-123	40	CRDNRC-123	INV555	40
CRDNRQ-456	100	CRDNRQ-456	INV555	10

Table 4–4

Credit Note ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNT-246	40	CRDNRC-123	INV555	40
CRDNT-369	10	CRDNRQ-456	INV555	10

The credit note request associated with a credit note is determined from the Ref No 3 field that should contain the credit note request Id. In the example, the total extended cost for a credit note matches exactly with the extended cost of its respective Credit Note Request. Therefore, all credit notes and credit note requests will be set to matched status.

Example 2: Quantity matching required by supplier, outside tolerance

The following example illustrates an unsuccessful match, where quantity matching is required by the supplier, and the tolerance level is set to 10%. It is assumed that the documents are matchable by credit note request ID.

Table 4–5

Credit Note Request ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNRC-123	40	CRDNRC-123	INV555	20
CRDNRC-456	100	CRDNRQ-456	INV555	2

Table 4–6

Credit Note ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNT-246	500	CRDNRC-123	INV555	25

In the example, the match is unsuccessful, despite the fact that the extended costs do match. The failed match is due to the requirement by the supplier to match quantities, and the difference in quantities on the credit note request--and the credit note is more than the allowed tolerance of 10 percent. The credit notes and credit note requests will remain in their original status.

Example 3 (quantity matching required by supplier, within tolerance)

The following example illustrates an unsuccessful match when the pool key is the credit_note_request_ID (Ref No 3), but a successful match when the pool key is Invoice_ID. In this scenario, quantity matching is required by the supplier, and tolerance level is set to 10.

Table 4–7

Credit Note Request ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNRC-123	40	CRDNRC-123	INV555	20
CRDNRC-456	100	CRDNRQ-456	INV555	4

Table 4–8

Credit Note ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNT-246	500	CRDNRC-123	INV555	25

In the example, the match is not successful when using Credit Note Request ID (reference field 3) as the pool key, because the extended cost difference (500 - 400) is outside of tolerance.

However, if the batch process was using the Invoice ID (reference field 4) as the pool key the match would be successful because the extended costs (400+100 = 500) match, and the quantities match within tolerance (20+4 = 24, where 24 is within 10% of 25). All the credit note requests and credit notes will be set to the status of matched. Example 4 illustrates a scenario in which Invoice ID is used as the pool key.

In case of cost discrepancies, the costs will match if the extended cost differences between the credit note request and the credit note are within tolerance.

Example 4: Matchable by invoice ID.

If the credit notes and credit requests are not matchable by the credit note request ID. The matching process will attempt a one-to-one, and then a line-level match before moving to the next document key-set which is the invoice ID. The invoice ID is populated in the Ref 4 field of the credit note. The following example illustrates an attempted summary match which failed when using credit note request ID, but is successful when the match is attempted using the second priority Pool Key attribute, Invoice ID.

Table 4–9

Credit Note Request ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNRC-123	20	CRDNRC-123	INV555	2
CRDNRC-456	80	CRDNRQ-456	INV555	8

Table 4–10

Credit Note ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNT-246	100		INV555	10

In the example, the Ref 3 field is empty. Therefore, a match attempted with the Credit Note Request ID fails. Assuming that one-to-one and live-level matches also fail, a second attempt to summary match will be made in the next document-key set (using the invoice ID). In this case, because the credit note and credit note requests match by Invoice ID, all credit notes and credit note requests will be set to matched status.

Example 5: Matchable by PO Location

If a credit note and credit request is not matchable in the document-key sets utilizing credit note request ID, or invoice ID, then the match will be attempted using the PO and Location combination which is the third priority in the default Pool Key Attributes of the system.

Assuming that the invoice and receipt data in the in application is as follows:

- Purchase Order=89890
- Location=1000001

Table 4–11

Document Type	Document ID	Unit Cost	Extended Cost	Quantity
Invoice	INV555	$11.00	$440	40
Receipt	SHP444	$10.00	$200	30

When matched, the invoice and receipt will create a cost discrepancy of $40 and a discrepancy of $100, which will generate a credit note request for $40 and a credit note request quantity for $100.

Table 4–12

Credit Note Request ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNRC-123	40	CRDNRC-123	INV555	40
CRDNRC-456	100	CRDNRQ-456	INV666	10

Table 4–13

Credit Note ID	Total Extended Cost	Ref No 3	Ref No 4	Quantity
CRDNT-246	140			50

In the example, both the Ref No 3 and Ref No 4 fields on the credit note are empty. Therefore matching is not even attempted at the Credit Note Request ID or invoice ID pool keys. A third attempt to match will be made with the PO and Location combination. As calculated from the above data, the PO and location combination for all three documents is: 89890-1000001. Since this combination is the same for all three documents, and the extended cost of the credit note requests match with the credit note, all credit notes and credit note requests will be set to matched status.

One-to-One Invoice Matching Algorithm

One-to-one credit note matching is considered another form of summary matching. The only addition to the rule is that instead of attempting matches in groups, one-to-one matching attempts to match a single credit note with a single credit note request.

The batch first attempts a match between the total extended costs. If the total extended costs do not match exactly for the credit note and the credit note request pair, then tolerances are applied to check if the cost discrepancy is within tolerance. In case quantity matching is required by the supplier, header level quantity matching is also

attempted for the document pair within tolerance. If no match can be found, the documents are left in their original status.

One-to-one algorithm will attempt a match only when at least one unmatched credit note exists in the document-key set. If no unmatched credit notes remain, then processing stops for the document-key set.

Some scenarios of one-to-one matching are listed below. Note that the examples are given to demonstrate an understanding of one-to-one matching algorithm. It is assumed that quantity matching is required in the supplier options, and documents are matchable only by credit Note Request ID.

Example 1: Exact Match

The following example illustrates how one credit note matches with one and only one credit note request. One credit note and two credit note requests are left in their original approved status.

Table 4–14

Credit Note ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNT 1	$50,000	$5,000	Matched
CRDNT 2	$100,000	$10,000	Approved

Table 4–15

Credit Note Request ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNRC 1	$50,000	5.000	Matched
CRDNRC 2	$25,000	2,500	Approved
CRDNRC 3	$35,000	2,500	Approved

In the example, CRDNT 1 matches with CRDNRC 1. However, the remaining credit (CRDNT 2) does not match with either of the two remaining credit note requests so it remains unmatched. The remaining credit note requests (CRDNRC 2 and CRDNRC 3) are also left in their original state. The matching algorithm will now move to the next matching algorithm within the document-key set to consider matching these documents.

Example 2: Match unsuccessful; one credit note but two credit note requests

The following example illustrates an unsuccessful match i.e. no successful matches.

Table 4–16

Credit Note ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNT 1	$50,000	5,000	Approved
CRDNT 2	$25,000	2,500	Approved
CRDNT 3	$35,000	3,000	Approved

Table 4–17

Credit Note Request ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNRC 1	$40,000	5,000	Approved
CRDNRC 2	$25,000	2,500	Approved
CRDNRC 3	$25,000	2,500	Approved
CRDNRC 4	$10,000	1,000	Approved

In the example, CRDNT 2 can be successfully matched to both CRDNRC 2, and CRDNRC 3. Therefore, no match can be obtained for Invoice 2. All credit notes and credit note requests are left in their original status.

Example 3: Match unsuccessful; two credit notes for one credit note request

The following example illustrates another multi-unresolved match, with no successful matches.

Table 4–18

Credit Note ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNT 1	$40,000	4,000	Approved
CRDNT 2	$25,000	2,500	Approved
CRDNT 3	$25,000	2,500	Approved
CRDNT 4	$10,000	1,000	Approved

Table 4–19

Credit Note Request ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNRC 1	$50,000	5,000	Approved
CRDNRC 2	$25,000	2,500	Approved
CRDNRC 3	$35,000	3,000	Approved

In the example, CRDNRC 2 can be successfully matched to both CRDNT 2 and CRDNT 3. All credit notes and credit note requests are left in the original Approved status.

Example 4: All credit notes are matched, but credit note requests remain.

The following example illustrates a scenario in which all credit notes match, but there are remaining unresolved credit note requests.

Table 4–20

Credit Note ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNT 1	$50,000	5,000	Matched
CRDNT 2	$25,000	2,500	Matched
CRDNT 3	$35,000	3,000	Matched

Table 4–21

Credit Note Request ID	Total Extended Cost	Total Quantity	Status Post Matching
CRDNRC 1	$50,000	5,000	Matched
CRDNRC 2	$25,000	2,500	Matched
CRDNRC 3	$15,000	2,500	Approved
CRDNRC 4	$35,000	3,000	Matched
CRDNRC 5	$75,000	10,000	Approved

In the example, all three credit notes are successfully matched to one and only one credit note request. However, two unmatched credit note requests remain. Since there are no credit notes left for matching, processing will stop for this document-key set.

Line Level Matching Algorithm

Once summary matching and one-to-one matching pools have been exhausted, the CreditNoteAutoMatchBatch proceeds to attempts match at the line level.

In addition to the eligibility requirements for summary matching, lines must be present on the documents for Line-level matching to proceed. The batch assumes that all lines are present and valid for that credit note and credit note request. Moreover, the algorithm attempts matches only if there is just one credit note left unmatched in the document-key set.

Considering that only one eligible credit note and zero-to-many credit note requests are unmatched for a document key-set, the system attempts to match each line item on that credit note to a credit note request line item. When a detail level match is found, the detail on the credit note and credit note request documents are both flagged as matched. Once line level matching is complete for a document key-set, and all lines have been matched, then the entire credit note and all of its related credit note requests are considered matched. Otherwise, they remain in their original approved status.

For line-level matching, cost and quantity matching are always performed. If cost matching fails, quantity matching is still performed in order to route potential quantity discrepancies that may be discovered.

For quantity line level matching, the comparison is made between the sum of quantities from the credit notes and sum of the quantities on the credit note request for that item. If a quantity match cannot be obtained, then a quantity discrepancy is generated and routed for the credit note and the credit note request lines for that item.

Example 1 (match within tolerance)

The following example illustrates a scenario in which all lines match within tolerance, and the credit notes and credit note requests are set to matched status.

Table 4–22

Credit Note	Item	Unit Cost	Quantity	Status Post Matching
CRDNT 1			550	Matched
Line	1	$5.00	100	Matched
Line	2	$10.00	200	Matched
Line	3	$15.00	250	Matched

Table 4–23

Credit Note Request	Item	Unit Cost	Quantity	Status Post Matching
CRDNRC			565	Matched
Line	1	$5.02	105	Matched
Line	2	$10.10	210	Matched
Line	3	$15.03	250	Matched

In the example, assume line-level tolerances are set such that all lines match, therefore the line-level statuses are set to matched accordingly.

Example 2 (match resolving discrepancy)

The following example illustrates a scenario in which tolerances allow only one line to matches, but the CreditNoteAutoMatchBatch is able to resolve the discrepancies in other items, and match the credit note and credit note request.

Table 4–24

Credit Note	Item	Unit Cost	Quantity	Status Post Matching
CRDNT 1			550	Matched
Line	1	$12.00	100	Matched (by resolving discrepancy)
Line	2	$10.00	200	Matched
Line	3	$12.00	250	Matched (by resolving discrepancy)

Table 4–25

Credit Note Request	Item	Unit Cost	Quantity	Status Post Matching
CRDNRC 1			600	Matched
Line	1	$12.00	110	Matched (by resolving discrepancy)
Line	2	$10.10	200	Matched
Line	3	$10.10	250	Matched (by resolving discrepancy)

In the example, the lines value for Item 2 is matched. However, Items 1 and 3 do not match within tolerance. If however, reason codes are entered in the appropriate default columns for automatically handling Credit Note matching discrepancies in the system options table, then discrepancies are created automatically for Item1 and Item 3 and the two items are set to matched status on both documents.

Discrepancy Creation and Resolution in Line Level Matching

When discrepancies are created as part of the line-level matching process, they are automatically resolved by the batch process. This resolution takes place by selecting the appropriate reason code from the system options and then resolving those discrepancy by creating resolution actions in the system. For instance, if a cost discrepancy is detected, then a resolution action in the form of a Credit Note Request

for cost or a Credit Memo is generated. On the next run of the reason code action rollup process, these newly created resolution actions will be rolled up to create the appropriate resolution documents.

It is important to distinguish the differences between overages that are in favor of the retailer as opposed to the supplier. In credit note matching, when a credit note is greater than the credit note request issued for it, the overage is in the favor of the retailer and a credit memo is issued to reconcile the discrepancy. This is because the credit note already represents an asset to the retailer. The supplier has issued more credit to the retailer than was appropriate based on the credit note request. If the retailer does not wish to automatically issue credit memos when the credit note is larger than the credit note request, then the system options, Auto-resolution Reason Code for Credit Memo - Cost and Auto-resolution Reason Code for Credit Memo-Qty, should be left blank.

If the applicable system option for a resolution action code type does not have a reason code defined in the System Options Maintenance screen then discrepancies of that type are not generated. It is assumed that the retailer will handle these discrepancies manually. This means that the credit note will not be matched and processing will stop for the document set. This allows for the retailer to have the batch resolve only specific discrepancy types. For example, many retailers may not want to automatically generate Credit Memos in response to Credit Note overages.

Example 1 (cost discrepancy)

The following example illustrates a scenario in which the first line on the credit note matches with the first line on the credit note request. The second line has a cost discrepancy.

Table 4–26

Credit Note	Item	Unit Cost	Quantity	Status Post Matching
CRDNT 1			300	Matched
Line	1	$12.00	100	Matched
Line	2	$5.00	200	Matched (by resolving discrepancy)

Table 4–27

Credit Note Request	Item	Unit Cost	Quantity	Status Post Matching
CRDNTR 1			600	Matched
Line	1	$12.00	110	Matched (by resolving discrepancy)
Line	2	$10.10	200	Matched
Line	3	$10.00	250	Matched (by resolving discrepancy)

In the above scenario Item 2 in credit note and credit note request has a cost discrepancy of $5. The Line level match algorithm automatically generates a cost discrepancy for the item, and generates a Resolution Action in the system for a Credit Note Request - Cost, where the total extended cost on the credit note request is $1,000.00.

Example 2 (quantitydiscrepancy)

The following example illustrates a scenario in which the first line on the credit note matches with the first line on the credit note request. The second line has a quantity discrepancy.

Table 4–28

Credit Note	Item	Unit Cost	Quantity	Status Post Matching
CRDNT 1			300	Matched
Line	1	$12.00	100	Matched
Line	2	$10.00	200	Matched (by resolving discrepancy)

Table 4–29

Credit Note Request	Item	Unit Cost	Quantity	Status Post Matching
CRDNTR 1			310	Matched
Line	1	$12.00	110	Matched (by resolving discrepancy)
Line	2	$10.10	210	Matched

In the above scenario, Item 2 in credit note and credit note request has a quantity discrepancy of 10 (assumed to be above tolerance values). The Line level match algorithm automatically generates a quantity discrepancy for the item, and generates a resolution action in the system for a Credit Note Request - Quantity. The above discrepancies are in the favor of the supplier. In case of the discrepancy being in the favor of the retailer, resolution actions would have been Credit Memos (cost or quantity).

Example 3 (orphan items - credit memo)

A case might exist when an item on the credit note does not exist on the credit note request. The CreditNoteAutoMatchBatch will resolve the discrepancy of the orphan items by utilizing the resolution actions for Credit Memos.

Table 4–30

Credit Note	Item	Unit Cost	Quantity	Status Post Matching
CRDNT 1			300	Matched
Line	1	$12.00	100	Matched
Line	2	$10.00	200	Matched (by resolving discrepancy)

Table 4–31

Credit Note Request	Item	Unit Cost	Quantity	Status Post Matching
CRDNTR 1			150	Matched
Line	1	$12.00	110	Matched

In this scenario Item 2 in credit note does not exist in the credit note request. This orphan item creates a cost discrepancy. Since the discrepancy is in the favor of the retailer, the Line level match algorithm will automatically generate a Resolution Actions in the system for a Credit Memo-Cost, where total cost on the memo is $2,000.00.

Role of Reason Code Action Rollup Batch in Credit Note Matching

The ReasonCodeActionRollupBatch facilitates the CreditNoteAutoMatchBatch process in the following ways:

- **Document Creation**

  Resolution actions created as part of the discrepancy creation process of the CreditNoteAutoMatchBatch are converted to first class documents on the next run of the ReasonCodeActionRollupBatch. This is an existing feature of the ReasonCodeActionRollupBatch.

- **Transfer of Customizable Reference Fields**

  Currently, documents in the system have a set of four (4) customizable reference fields. These fields are completely optional and their population is left to the discretion of the retailer and their suppliers. If applicable, the CreditNoteAutoMatchBatch utilizes the third and fourth customizable reference fields to facilitate matching. In such cases the ReasonCodeActionRollupBatch makes sure that the values of those fields are transferred to the newly created documents.

- **Tolerances**

  Tolerances are handled in a manner similar to the invoice auto match batch process, and tolerance values used for credit note auto match are same as the tolerance values used for invoice matching.

  Matching tolerances are defined at the following levels:

  – Summary or Line

  – Cost or Quantity

  – Favor of Retailer or Supplier

  – Amount or Percentage

  Summary matching and one to one matching are both considered types of summary matching, therefore, one to one matches also uses summary level tolerances.

  During the match process, tolerances are selected in the following order:

  1. Supplier
  2. Department
  3. System

  If no supplier level tolerances are defined, then departmental tolerances are used for a random item in the document set. If departmental tolerances are not defined for that item, then system level tolerances are used for the document set. Note that a document set must have items to use department level tolerances.

Currencies

Tolerance values are stored using the primary system currency. If the credit note currency and the credit note request both have the same currency but that currency differs from the system currency, the CreditNoteAutoMatchBatch will convert them into the system currency utilizing the currency APIs to determine if the documents fall within appropriate tolerances. Note that the CreditNoteAutoMatchBatch process will not attempt to match a credit note with a credit note request if the currency between the two documents is not the same.

TAX Matching

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 for each item the Tax codes and rates 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 this circumstance, the Invoice Matching user will have to match and resolve the Tax discrepancy manually through the user interface.

History and Record Keeping

ReIM records summary and detail history for matched credit notes and credit note requests. The existing credit note matching history data model will be leveraged to ensure than an accurate accounting of match data is stored in the system. In addition, a new history data model has been introduced which holds history data for a specific match. The new history tables are populated after the completion and success of a match.

Data Purge

When document data becomes dated, it is purged from the system through the TablesPurgeBatch. This is also true for the CreditNoteAutoMatchBatch related data. See "Batch Processes" in Chapter 9, "Batch Processes," for information.

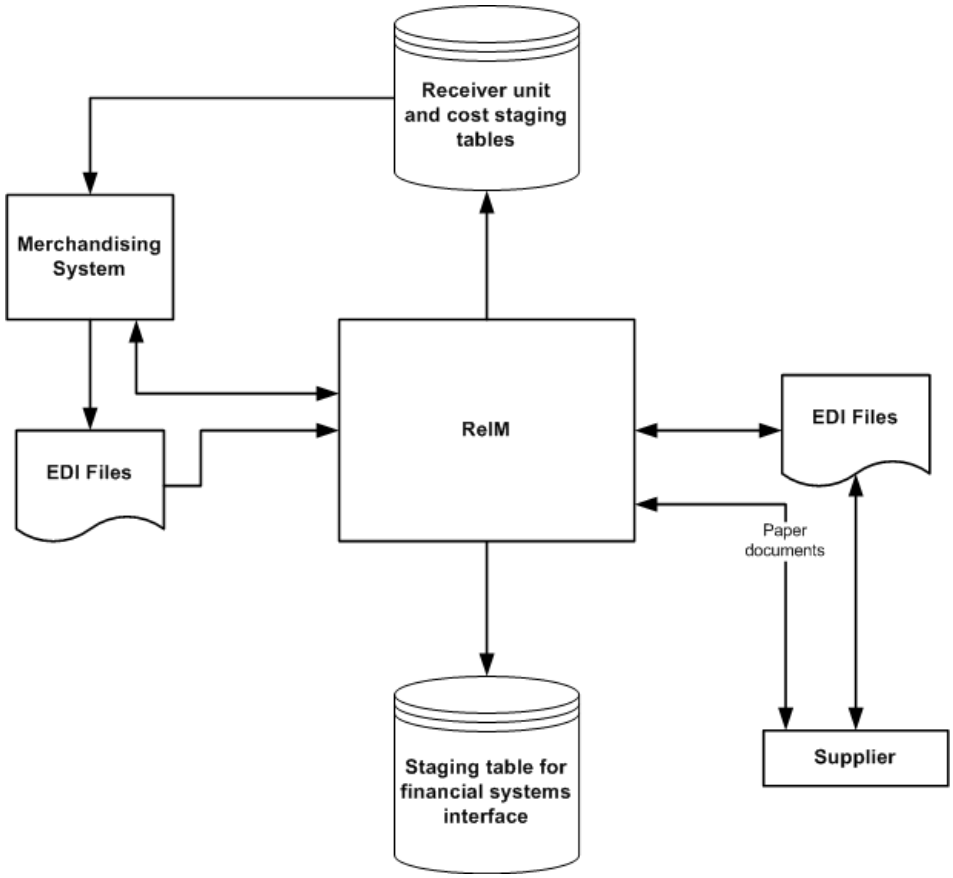
5

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.

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.

- **Supplier Trait**

  An RMS function, supplier traits are used as a grouping mechanism for suppliers with common characteristics. They are utilized for mass updates. This data is used in setting up tolerances within ReIM.

- **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 9, "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 (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. Multi-location invoices will be inserted into IM_PARENT_INVOICE, IM_PARENT_INVOICE_DETAIL and IM_PARENT_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. Reject transaction to tables if: ■ Vendor is a supplier and supplier is not valid. ■ Vendor is a supplier and vendor ID is not completely numeric.
Vendor ID	Char(10)	Vendor for this document. Stored in IM_DOC_ HEAD.VENDOR_TYPE	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).
Vendor Document Date	Char(14)	Date document was issued by the vendor (in YYYYMMDDHH24MISS format). Stored in IM_ DOC_HEAD.DOC_DATE	Y	

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.
Multi Location	Char(1)	Indicates if this invoice goes to multiple locations. If Yes, the record should be inserted to IM_PARENT_ INVOICE table.	Y	Reject transaction to file if: ■ Multi-location is null. ■ Multi-location is not Y or N. ■ Multi-location is Y and Consignment is Y.
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.

Field Name	Field Type	Description	Req	Validation
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 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(30)	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(30)	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(30)	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(30)	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

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.

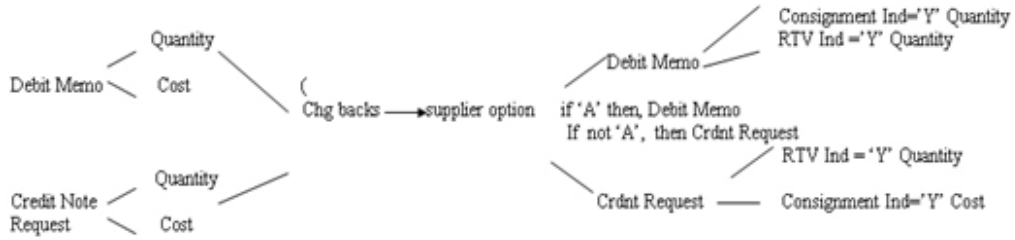
Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	Describes file record type.	Y	FTAIL

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	
Group Id	Char(10)	Invoice Group ID Retrieved from IM_DOC_HEAD.GROUP_ID.	Y	
Invoice Number	Char(10)	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	

Field Name	Field Type	Description	Req	Validation
Location	Number(10)	Location for this document, if any. Retrieved from IM_DOC_ HEAD.LOCATION.	N	
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	

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

To be independent of any single financial product, such as Oracle Financials, Oracle retail has created a generic interface. That is, Oracle Retail writes records to a single generic table from which custom retailer code can read records and process data as necessary. The retailer is responsible for creating a process that sends transactions to the financials system.

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" in Chapter 9.

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.

For information, see "Authentication Settings" in Chapter 2, Baclemd System Administration and Configuration."

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

Parameter	Description
user_first_name_attribute_name-	Name for the first name attribute used within the attribute class
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.

6

Technical Design

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

Locking Design Summary

ReIM locking is accomplished using database tables that hold record level locks. The locking of tables is performed for several reasons, including the following:

- ReIM does not necessarily maintain a single connection throughout an entire screen/process. That is, the system opens a connection, fetches information, and then closes the connection. At a later moment in time, the system opens another connection to save changes and close the connection.
- ReIM cannot maintain locks in some kinds of Java session structures because the system may be involved with more than one Java virtual machine (JVM).

Locking and Tables

Base tables that contain information to be locked (for example, IM_SUPPLIER_OPTIONS) have a corresponding …_LOCK table (for example, IM_SUPPLIER_OPTIONS_LOCK). The …_LOCK table contains the same columns as the primary key of the base table.

When the system creates a lock, it writes the primary key values for the base table records to be locked to the appropriate …_LOCK table. For example, if data in the IM_SUPPLIER_OPTIONS table is to be locked for supplier 12345, a record is written to the IM_SUPPLIER_OPTIONS_LOCK table for supplier with the primary key value, 12345.

When records in a base header table are locked, all detail records related to each locked header record are implicitly locked. Detail records are not explicitly locked because:

- ReIM functionality must go through the header information to access detail information. In other words, the entry point to detail records is generally through the header.
- On screens and within backend processes that include header information, some kind of summary of the details also exists.

The following two examples represent this type of header detail locking:

Example 1

If user A is looking at the header, and user B changes the details, user A does not have visibility to the changes and might perform an invalid action. Invoices are stored on IM_DOC_HEAD, and the non-merchandise costs on invoices are stored on IM_DOC_NON_MERCH. On the invoice header screen, user A can see a sum of all of the non-merch costs for invoice 99999. If user B could somehow at the same time add new

non-merchandise costs for invoice 99999, the information that user A sees as the summary of non-merchandise costs would be invalid.

Example 2

If auto-match has selected all documents 'ready for match' and is processing and then additional data is entered for a document, the details with which the auto-match is working would no longer be valid.

Locking Management

For locking management, consider the following.

- When a user that has an active lock exits a screen (that is, the user selects **OK** or **Cancel** buttons on the screen), data changes are committed (if necessary) and then any locks on data displayed on that screen are removed. If any expired locks on the screen data exist, they are also released upon screen exit.
- When a user tries to commit information to the database, the locking service checks to ensure that the user has valid locks on any changed data being committed (for example, locks could have timed out as noted below). If the user does not have valid locks, the user receives a message noting that the user's changes cannot be saved. In this case, the user must exit the screen, enter the screen again, and re-enter the data changes that could not be committed due to invalid/expired locks.
- In situations where accidental system exits occur (for example, the server shuts down unexpectedly from power loss), locks are not released immediately. After the system is restored from outage, the user will log into the system and access the main menu. At that point, any existing data locks are removed. Because this data is no longer locked, any user with adequate security permissions can acquire new locks on this data.
- The lock timeout interval is defined in the reim.properties file. See Chapter 2, "Backend System Administration and Configuration," for information.
- When locks are written to the …_LOCK table, they include an 'end time' value. When checking to see if a row of data is locked, the system inspects the related lock row 'end time' value. If the commit time is before the end time on the …_ LOCK table record, the base table data changes may be committed. If the commit time is equal to or exceeds the end time, the data lock will be treated as 'expired' and the data changes will not be committed.
- If a user needs immediate access to already locked data and cannot wait for data locks to expire or be released by the user holding the locks, a database administrator can manually delete existing lock records from the appropriate …_ LOCK table to release the locks. However, this does not guarantee that the user that needs immediate access will be the next user to acquire locks on the just-released data. The manual release of locks should be a rare event due to the other lock release methods in the system.

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 primary currency of the system. 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 primary currency of the system. 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-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 (both single invoice entry and batch 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.

Java Currency Formatting

Currency must be properly formatted according to its applicable locale. For example, US currency uses a comma as a thousands separator whereas other currencies do not use a comma as a thousands separator. Java has built-in libraries for currency formatting that are based on locales.

ReIM uses built-in Java localization functionality mapped through the table IM_CURRENCY_LOCALE to RMS existing currency structure. ReIM provides an installation script that populates this table. The script creates records for every currency that RMS supports. Note that ReIM cannot guarantee the accuracy of RMS language data.

7

Oracle E-Business Suite Financials Integration using Oracle Retail Financial Integration

This chapter describes the integration between Oracle Retail systems and Oracle E-Business Suite Financials (including Oracle General Ledger and Oracle Payables), as developed and supported by Oracle Retail Financial Integration (ORFI).

When the option to integrate is chosen, the selected information is shared between the systems. Integration and validation services are in place to ensure the shared data matches.

Note: This chapter addresses the points within Oracle Retail systems that are essential to integration. For more information about the entire integration process, including mapping to Oracle E-Business Suite data and settings, see the ORFI document, *Oracle Retail Financial Integration for Oracle Retail Merchandising Suite and Oracle E-Business Suite Financials - Implementation Guide*. For more information about Web services, see the following chapters in the *Oracle Retail Merchandising System Operations Guide, Volume 2:* "Service Provider Implementations API Designs" and "Web Services."

Participating Applications

The following Oracle Retail applications are included in the integration covered by this chapter:

- Oracle Retail Merchandising System (RMS)
- Oracle Retail Sales Audit (ReSA)
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Integration Bus (RIB)

Assumptions and Dependencies

- The option to integrate must be selected during initial set up of the RMS system.
- ReIM accesses RMS to determine if integration is active. Initial set up of RMS must occur prior to the integration of ReIM.

- The URLs for the RFI Web services that are a necessary for this integration must be maintained in the RMS_RETAIL_SERVICE_REPORT_URL table and in the ReIM integration.properties file.
- Real time account validation is done only when the financial integration with Oracle E-Business Suite is ON.
- Partners must be set up as suppliers in Oracle E-Business Suite. Then the partner must be manually set up in RMS using the RMS Supplier ID that was generated when the Oracle E-Business Suite supplier was interfaced to Oracle Retail. Partner functionality within RMS and ReIM can then proceed normally. The RMS supplier generated as part of this process is not used.
- Payment terms and freight terms are manually maintained.

Data Setup

Integration of Oracle Retail applications and Oracle E-Business Suite Financials relies on synchronization of essential data, such as currency exchange rates and suppliers. Through careful discussions, the users of both systems determine the common codes and descriptions that will best serve their business needs.

Once agreement is reached, this information is set up and maintained. Depending on the volume, some shared information is set up in Oracle Retail applications or in Oracle E-Business Suite and electronically transferred to the other systems. Otherwise, shared information is set up manually within each system, and the user of both systems must ensure that the code and the description match.

RMS Data Setup and Configuration

This section describes setup considerations for the RMS data.

RMS System Options

As part of the RMS system options setup, set the following options as indicated:

- FINANCIAL_IND=Y

This system_option indicates that the Oracle Retail system is integrated with a financial system:

- FINANCIAL_AP=A

A value of A indicates that the financial system to which RMS is interfaced is Oracle E-Business Suite through Oracle Retail Financial Integration (ORFI).

- GL_ROLL_UP can be D/S/C
- MULTIPLE_SET_OF_BOOKS_IND = Y
- SUPPLIER_SITE_IND = Y
- ORG_UNIT_IND = Y

Organization Units

Use the Organizational Unit window (RMS Start Menu > Control > Setup > Org Unit >Edit) to define organizational units in RMS that match those being setup in Oracle E-Business Suite. When an organizational unit is entered in RMS, the valid organizational units are those associated with the Set Of Books (SOB) that is being used for the general ledger interface.

Currency Exchange Rates

Currency exchange rate is used to translate the monetary value of one currency in terms of another. Depending on business needs, a Currency Exchange Rate Type of Operational or Consolidation is selected for use in all transactions.

This value is set up manually in RMS and mapped to Oracle E-Business Suite through the Currency Exchange Type mapping window. Currency Exchange Rate data is owned by Oracle General Ledger and updates are sent to Oracle Retail applications.

Determine the Exchange Type being sent by Oracle General Ledger (for example, Consolidation or Operational) that you want RMS to use. Update the FIF_CURRENCY_XREF for mapping the external exchange type being sent by Oracle General Ledger with RMS Exchange Type.

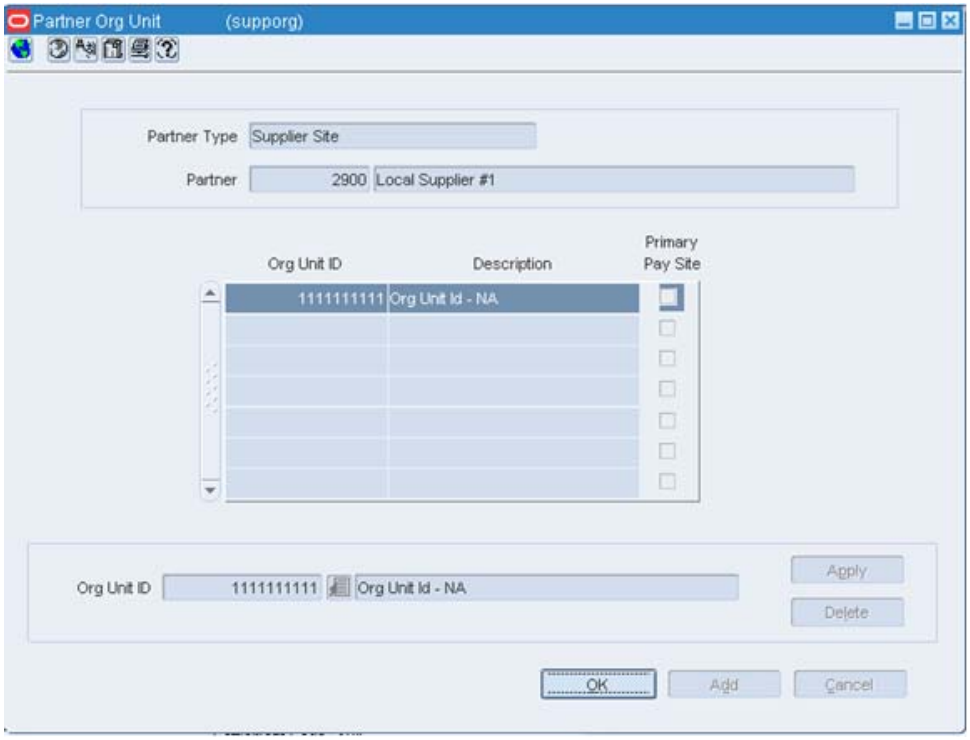
For example, for Consolidation and Operational exchange types, the FIF_CURRENCY_XREF table holds the following entries:

Table 7–1

FIF_EXCHANGE_TYPE	RMS_EXCHANGE_TYPE
C	C
O	O

Supplier Address Types

Within RMS, supplier information (such as Order From and Remit To addresses) is used for generating the purchase orders. Oracle Payables uses supplier information for payment generation. It is important that this information is synchronized.



Suppliers are created in Oracle Payables and exported to RMS. When FINANCIAL_AP is set to A, suppliers cannot be created using the RMS forms. However, after the supplier exists in RMS, all data values for the supplier (except supplier name and

status) continue to be updated using the RMS forms. The association of supplier sites to organization units is accessed only in view mode through RMS forms. One supplier site per supplier organization unit combination can be marked as primary payment site.

Where SYSTEM_OPTIONS.FINANCIAL_AP is A, disable auto generate supplier/partner numbers and associated check boxes.

Note: Supplier information is created, updated and inactivated only in Oracle Payables. This information is transferred from Oracle Payables to the participating Oracle Retail applications, where additional retail-specific attributes may be maintained.

Country Codes

When country codes are defined and seeded in RMS, ensure that country codes are mapped to Oracle E-Business Suite country codes through RFI DVM mapping. The following is an example of RFI DVM Mapping (Table RFI_XREF_DVM, available in RFI schema in Retail.

Table 7–2

EXT_SYSTEM_ID	COMMON_ID	RETL_ID
USA	700	US
CAN	701	CA

Financial Calendar

The financial calendar within Oracle Retail systems is manually set up and maintained separately from the Oracle General Ledger financial calendar.

Freight Terms

A freight term is an agreement between the retailer and a supplier regarding transportation charges for goods delivered by the supplier. Freight terms are used by RMS as purchase orders are generated.

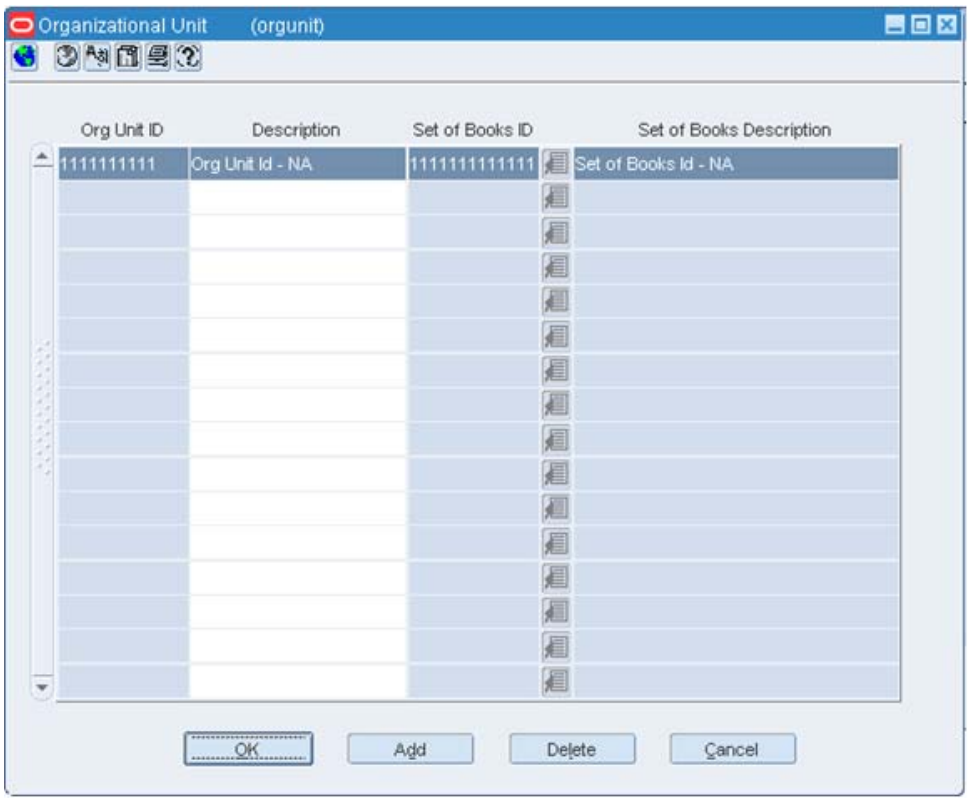
Within the RMS system, freight terms are set up and maintained manually. They also are maintained in Oracle Payables.

Payment Terms and Currency Exchange Rates

Currency exchange rates are created and updated in Oracle General Ledger and exported to RMS. Changes to Retail currency exchange rates are not propagated to Oracle General Ledger. Payment terms, however, are manually set up and maintained in each system.

Oracle E-Business Suite Financials Units and Site IDs

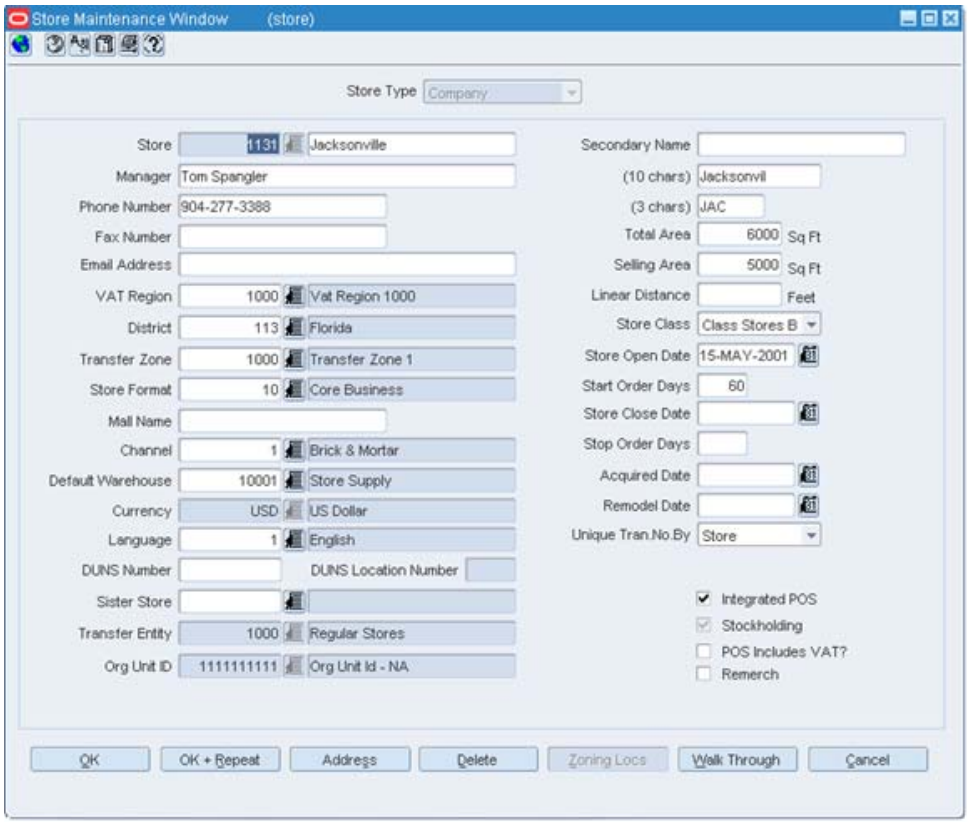
The data concepts of Org Units and Site IDs in RMS mirror the data maintained in Oracle E-Business Suite. RMS forms are used to manage and view Oracle Org Units and Site IDs. The RMS windows for Store and Warehouse maintenance allows for the association of each store and warehouse with an Org Unit. The following is an example of the Organizational Unit form:

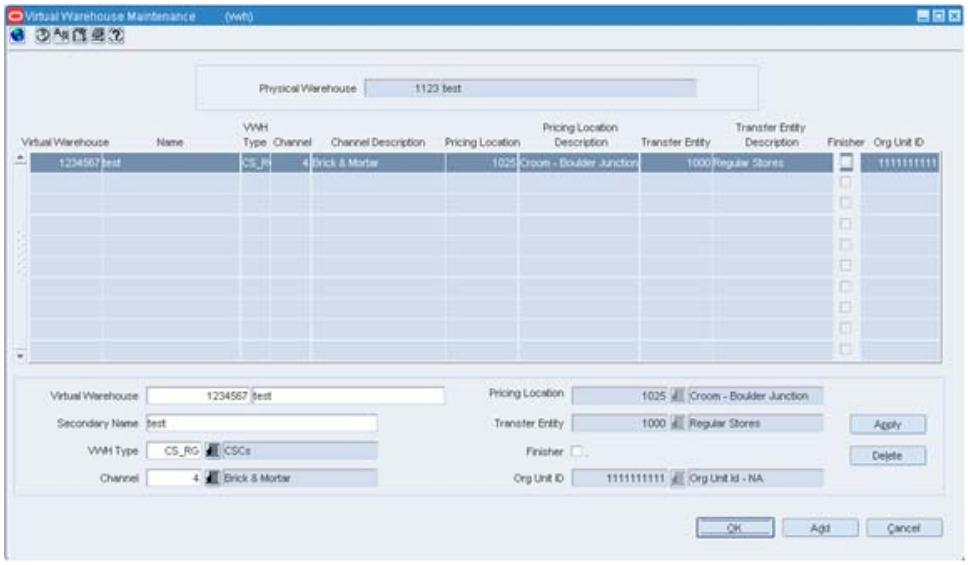


Store and Warehouse Maintenance

The organizational unit is found in the Store Maintenance and Warehouse forms, which allow the Oracle E-Business Suite operating unit to be associated with the Store or Warehouse. When RMS is set up for single-channel operation, the organizational unit is set at the physical warehouse level. When RMS is set up for multi-channel operation, the organizational unit is set up at the virtual warehouse level. Financial sales audit and inventory information can then be identified through interface routines and posted to the appropriate general ledger accounts. An organizational unit must be designated for each Store and Warehouse location in the RMS.

The following are examples of the Store Maintenance and Warehouse Maintenance forms:





RMS General Ledger Setup

For RMS and ReSA, manual setup is required for validating the chart of accounts. Valid chart of accounts are created and stored in general ledger cross reference tables. Once the validation is completed, transaction data can be assigned to specific account codes.

Ongoing maintenance of the chart of accounts information (such as adding, changing, or deleting chart of accounts) requires re-validation. In this regard, Oracle General

Ledger is the system of record, as it is used to verify the chart of accounts used by Oracle Retail applications. When these applications send a chart of accounts for validation, Oracle General Ledgers issues a message with:

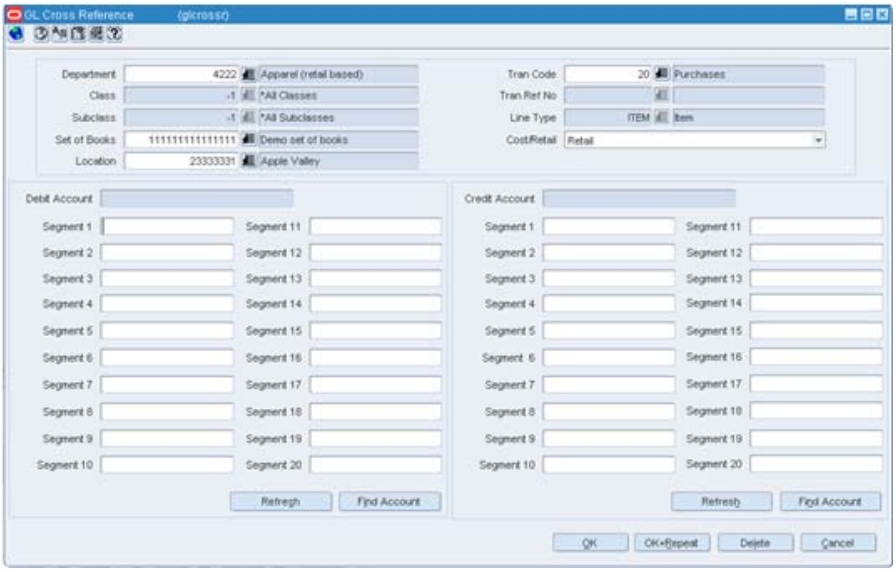
- Valid or invalid status
- Response date
- Chart of accounts

The RMS table, FIF_GL_SETUP, stores the Oracle E-Business Suite Set of Books IDs to post financial information. This table must be setup manually after Set of Books IDs are determined. Where system indicator Multiple Set of Books ID is set to N, FIF_GL_SETUP must hold a single Set of Books (SOB) record.

The Set of Books IDs is associated with the chart of accounts when setting up general ledger cross reference.

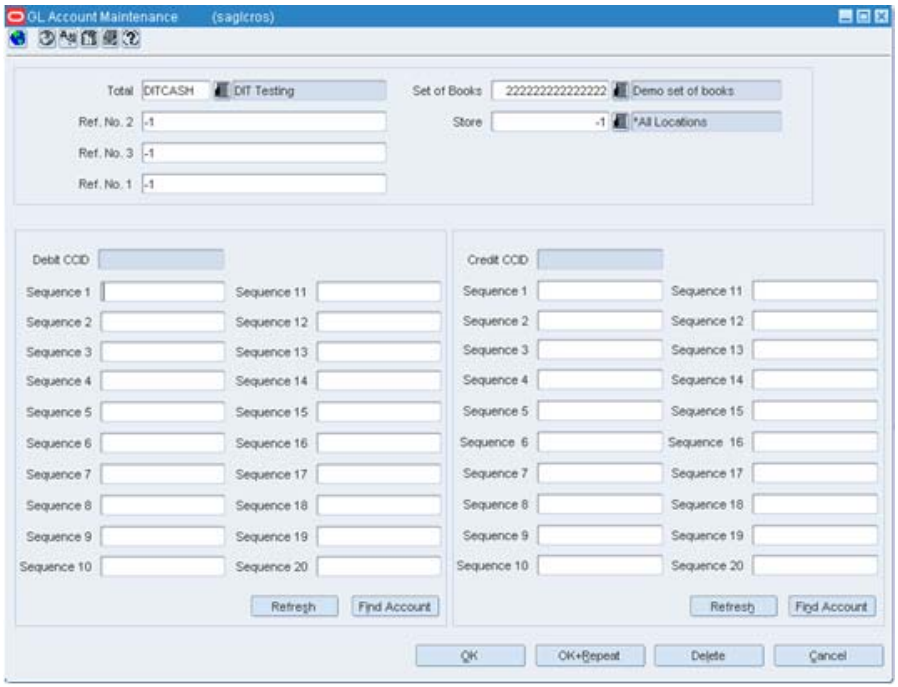
RMS General Ledger Cross Reference

Navigate: RMS Start Menu > Finance> GL Cross Reference. The General Ledger Search window opens. Map Chart of Accounts to department, Class, Subclass, Set Of Books, location, and transaction codes using the GL cross reference form in RMS.



ReSA General Ledger Cross Reference

Navigate: ReSA main menu > Action > Sales Audit > Control > Setup > GL Account Maintenance. The General Ledger Search Form window opens. Where SYSTEM_OPTIONS.FINANCIAL_AP is A, the form requires the entry of valid segment combinations.



ReIM Data Setup and Configuration

This section describes setup considerations for ReIM data.

System Options

As part of the RMS system options setup script, set the following options as indicated:

- FINANCIAL_IND =Y
- FINANCIAL_AP =A

As part of the ReIM system options setup script, DEFAULT_PAY_NOW_TERMS must be updated with the default term ID.

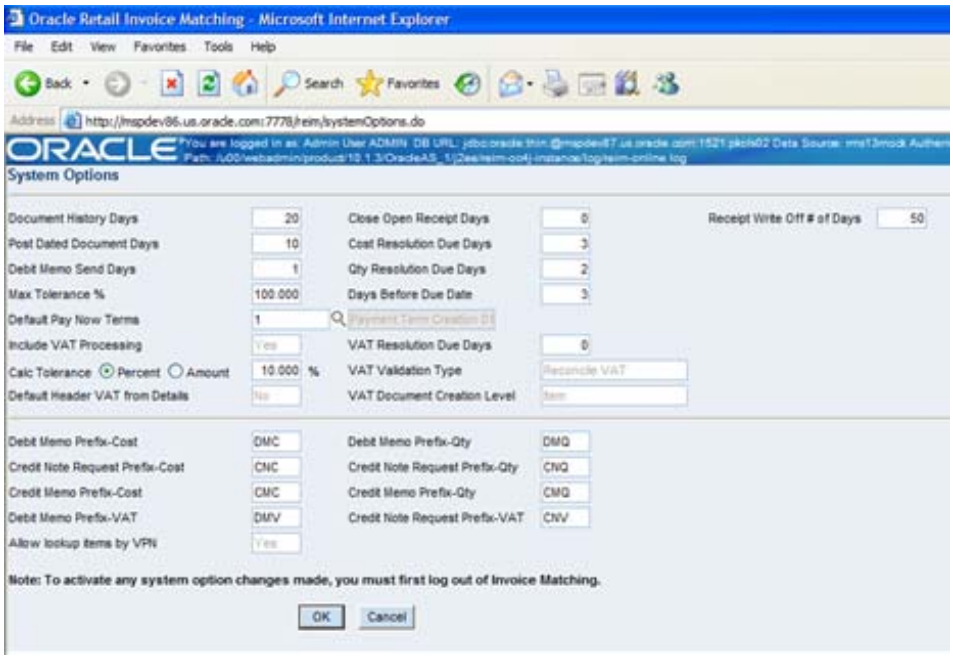
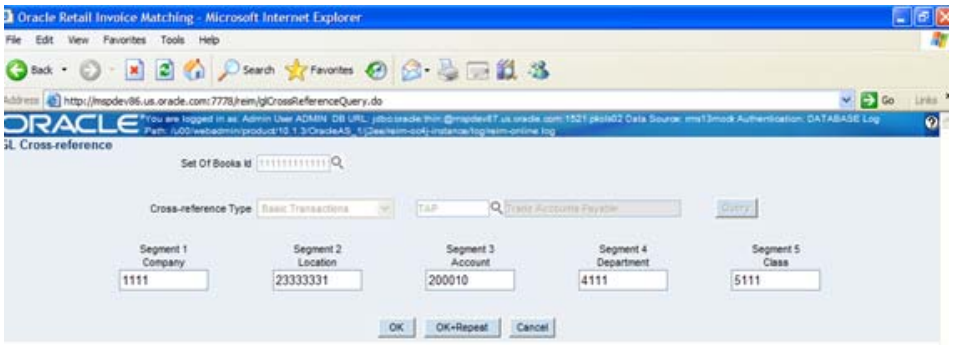


Chart of Accounts Setup

The chart of accounts is set up manually in Oracle Retail applications and in Oracle General Ledger. All account combinations are set up in each Set Of Books. The following is an example of the GL Cross Reference screen:



Note: The Chart of accounts is updated in Oracle Retail applications only after the account is validated through Oracle General Ledger.

Segment Mapping

The retailer determines how many segments are populated. Up to 20 account segments can be specified. The following is an example of how segments are mapped between the ReIM transaction table and Oracle General Ledger:

Table 7–3

ReIM Segments	Oracle General Ledger Chart of Accounts
Segment 1	PRODUCT

Table 7–3 (Cont.)

ReIM Segments	Oracle General Ledger Chart of Accounts
Segment 2	ACCOUNT
Segment 3	ALTACCT
Segment 4	OPERATING_UNIT
Segment 5	FUND_CODE
Segment 6	DEPTID
Segment 7	PROGRAM_CODE
Segment 8	CLASS_FLD
Segment 9	BUDGET_REF
Segment 10	BUSINESS_UNIT_PC
Segment 11	PROJECT_ID
Segment 12	ACTIVITY_ID
Segment 13	RESOURCE_TYPE
Segment 14	RESOURCE_CATEGORY
Segment 15	RESOURCE_SUB_CAT
Segment 16	CHARTFIELD1
Segment 17	CHARTFIELD2
Segment 18	CHARTFIELD3
Segment 19	AFFILIATE
Segment 20	AFFILIATE_INTRA1

If any one of the values in the 20 segments does not match the Oracle General Ledger, the account combination is considered as invalid. The following error message is issued to the user: "Account combination is invalid in the financial system."

Segments 1 and 2 may be set up as dynamic at the Location level, or Segments 4 and 5 can be dynamic at the Department and Class level respectively. Segments defined as dynamic are allowed to be null for certain types of Basic Transaction or Reason Code cross-reference types. When a segment is null, the segment is assigned dynamically when transactions are posted. (Non-dynamic segments cannot be blank.) Validation applies to the segment combination, not to individual segments.

Note: For Tran code TAP, each segment must have a value regardless of whether the segment is dynamic.

Running the Initial Load from Oracle E-Business Suite Financials

The initial load for ReIM is run by Oracle E-Business Suite and includes the following information:

- Suppliers
- Currency Rates

Note: The view, mv_currency_conversion_rates should be refreshed once the initial loads of currencies from Oracle General Ledger are loaded to ReIM.

integration.properties File Setup

To accommodate integration, the integration.properties file within ReIM must be updated with the appropriate URLs for the account validation and drill forward Web services, as listed below:

```
#webservice WSDL URL for drill forward
webservice.financial.drill.forward.wsdl=@webservice.drill.forward.wsdl@
 webservice.financial.drill.forward.url.targetnamespace=
webservice.financial.drill.forward.targetsystem=
#webservice WSDL URL for account validation
webservice.financial.account.validation=@webservice.account.validation@

webservice.financial.account.validation.namespace=http://www.oracle.com/retail/fin
/integration/services/GlAccountValidationService/v1

webservice.financial.account.validation.local.code=GlAccountValidationService
#webservice username and password for account validation
webservice.financial.account.validation.username=@webservice.account.validation.us
ername@
webservice.financial.account.validation.password=@webservice.account.validation.pa
ssword
@
```

Reports are created by Business Intelligence Publisher for the following:

The URL for each report must be updated in the table, retail_service_report_url. The following table provides sample URLs.

ReIM Transactional Maintenance

Integration to Oracle General Ledger includes a number of transactions, as described below.

Calculation of TRANS_AMOUNT

The TRANS_AMOUNT field in the im_financial_stage table stores the value of the journal entry to be posted to Oracle General Ledger. (The currency for the calculated amount is the currency assigned to the transaction.) The TRANS_AMOUNT value is calculated as follows:

Table 7–4

Row Description	DEBIT_CREDIT_IND	TRANS_AMOUNT Value
Normal	Debit	Transaction Amount
Normal	Credit	(-1) * Transaction Amount
VAT	Debit	Transaction Amount * Tax Rate
VAT	Credit	(-1) * Transaction Amount * Tax Rate

Note: Transaction Amount is taken from the database column, IM_FINANCIALS_STAGE.AMOUNT.

Generation of Outgoing Data

A staging table accommodates the outgoing transfer of data. The reference key assigned to each document or receipt is used to find data on this table.

Table 7–5

From	To	Transactions
ReIM	Oracle Payables	■ Invoices ■ Debit Memos ■ Credit Memos ■ Credit Notes
ReIM	Oracle General Ledger	General Ledger accounting entries resulting from the Invoice Matching process, including: ■ Pre-paid invoices ■ Receipt Write-offs
RMS	Oracle General Ledger	Accounting entry data (potentially very high volume)
ReSA	Oracle General Ledger	Accounting entry data (potentially very high volume)

Validation of Accounts When Posting Financial Entries

Valid chart of accounts are stored in the ReIM table, IM_VALID_ACCOUNTS, which includes the Set of Books ID (sob_id) and 20 segments. An ORFI Web service validates accounts against the Oracle General Ledger. Valid accounts are posted to IM_VALID_ACCOUNTS; invalid accounts are posted to IM_POSTING_DOC_ERROR. The following steps describe the validation process:

1. The ReIM system invokes the validation Web service to validate the chart of accounts. (A URL for the ORFI Web service is configured in the integration.properties file.)
2. The posting batch job checks the accounts to be posted against the IM_VALID_ACCOUNTS table.
3. If the chart of accounts is in the table, the transaction is posted to staging tables.
4. If the chart of account does not exist in the table, a collection of accounts is built. These collected accounts are validated against the Oracle General Ledger, and a status is returned.
    - If the status of the collected accounts is valid, the accounts are inserted in the IM_VALID_ACCOUNTS table, and the transactions are posted to the staging tables.
    - If the status of the accounts is NOT valid, the entire collection is flagged as errors, and transactions are posted to IM_POSTING_DOC_ERROR.

Note: ReIM completes the first level of account validation and posts the transaction to staging tables. It is assumed the second level of account validation is done at the end of the extraction process (where transactions are moved from ReIM staging tables to Oracle General Ledger). If account validation fails at this point, Oracle General Ledger must change the account information before transactions are loaded to Oracle General Ledger, and the chart of accounts must be re-validated in ReIM.

Maintenance of Valid Accounts

As account information is changed in the Oracle General Ledger, Retail must re-validate the locally stored chart of accounts. Oracle General Ledger will not propagate chart of account changes to Retail. The AccountPurge Batch can clear all valid accounts in the IM_VALID_ACCOUNTS table or only those that are considered updates in Oracle E-Business Suite.

Usage

```
reimaccountworkspacepurge batch-alias-name PURGE [ALL | <Accounts>]
```

Where

1. The argument is batch alias name.
2. The argument is the word PURGE.
3. The argument is either ALL or specific accounts to be deleted from the local table.

8

PeopleSoft Financials Integration using Oracle Retail Financial Integration

This chapter describes the integration between Oracle Retail systems and PeopleSoft Financials (including PeopleSoft General Ledger and PeopleSoft Payables), as developed and supported by Oracle Retail Financial Integration (ORFI).

When the option to integrate is chosen, the selected information is shared between the systems. Integration and validation services are in place to ensure the shared data matches.

Note: This chapter addresses the points within Oracle Retail systems that are essential to integration. For more information about the entire integration process, including mapping to data and settings, see the ORFI document, *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management and Oracle Financials Implementation Guide*. For more information about Web services, see the following chapters in the *Oracle Retail Merchandising System Operations Guide, Volume 2:* "Service Provider Implementations API Designs" and "Web Services."

Participating Applications

The following Oracle Retail applications are included in the integration covered by this chapter:

- Oracle Retail Merchandising System (RMS)
- Oracle Retail Sales Audit (ReSA)
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Integration Bus (RIB)

Assumptions and Dependencies

- The option to integrate must be selected during initial set up of the RMS system.
- ReIM accesses RMS to determine if integration is active. Initial set up of RMS must occur prior to the integration of ReIM.
- The URLs for the RFI Web services that are a necessary for this integration must be maintained in the RETAIL_SERVICE_REPORT_URL table and in the ReIM integration.properties file.

- Real time account validation is done only when the financial integration with PeopleSoft Financials is ON.
- Partners must be set up as suppliers in PeopleSoft. Then the partner must be manually set up in RMS using the RMS Supplier ID that was generated when the PeopleSoft supplier was interfaced to Oracle Retail. Partner functionality within RMS and ReIM can then proceed normally. The RMS supplier generated as part of this process is not used.
- Payment terms are automatically maintained.
- Freight terms are manually maintained.

Data Constraints

- The Location ID field is restricted to eight characters, to accommodate PeopleSoft Operating Unit, which has a maximum of eight characters.
- The Ext_Doc_ID field is restricted to 30 characters, because the corresponding PeopleSoft field has only 30 characters. Characters beyond 30 are truncated.
- RMS allows for four decimals, and PeopleSoft allows only three. Truncation may occur when data is passed to PeopleSoft Enterprise Financials.
- ReIM values in the IM_CURRENCY_LOCALE are restricted to three decimals, because the corresponding PeopleSoft Enterprise Financials field can accept no more than three decimal positions.

Data Setup

Integration of Oracle Retail applications and PeopleSoft Financials relies on synchronization of essential data, such as currency exchange rates, suppliers, and payment terns. Through careful discussions, the users of both systems determine the common codes and descriptions that will best serve their business needs.

Once agreement is reached, this information is set up and maintained. Depending on the volume, some shared information is set up in Oracle Retail applications or in PeopleSoft and electronically transferred to the other systems. Otherwise, shared information is set up manually within each system, and the user of both systems must ensure that the code and the description match.

RMS Data Setup and Configuration

This section describes setup considerations for the RMS data.

This system_option indicates that the Oracle Retail system is integrated with a financial system:

- FINANCIAL_AP=A

A value of A indicates that the financial system to which RMS is interfaced is PeopleSoft through Oracle Retail Financial Integration (ORFI).

- GL_ROLL_UP can be D/S/C
- SUPPLIER_SITE_IND = Y
- ORG_UNIT_IND = Y

Organization Units

Use the Organizational Unit window (RMS Start Menu > Control > Setup > Org Unit >Edit) to define organizational units in RMS that match those being setup in PeopleSoft. When an organizational unit is entered in RMS, the valid organizational units are those associated with the Set Of Books (SOB) that is being used for the general ledger interface.

Currency Exchange Rates

Currency exchange rate is used to translate the monetary value of one currency in terms of another. Depending on business needs, a Currency Exchange Rate Type of Operational or Consolidation is selected for use in all transactions.

This value is set up manually in RMS and mapped to PeopleSoft through the Currency Exchange Type mapping window. Currency Exchange Rate data is owned by PeopleSoft General Ledger and updates are sent to Oracle Retail applications.

Determine the Exchange Type being sent by PeopleSoft General Ledger (for example, Consolidation or Operational) that you want RMS to use. Update the FIF_CURRENCY_XREF for mapping the external exchange type being sent by PeopleSoft General Ledger with RMS Exchange Type.

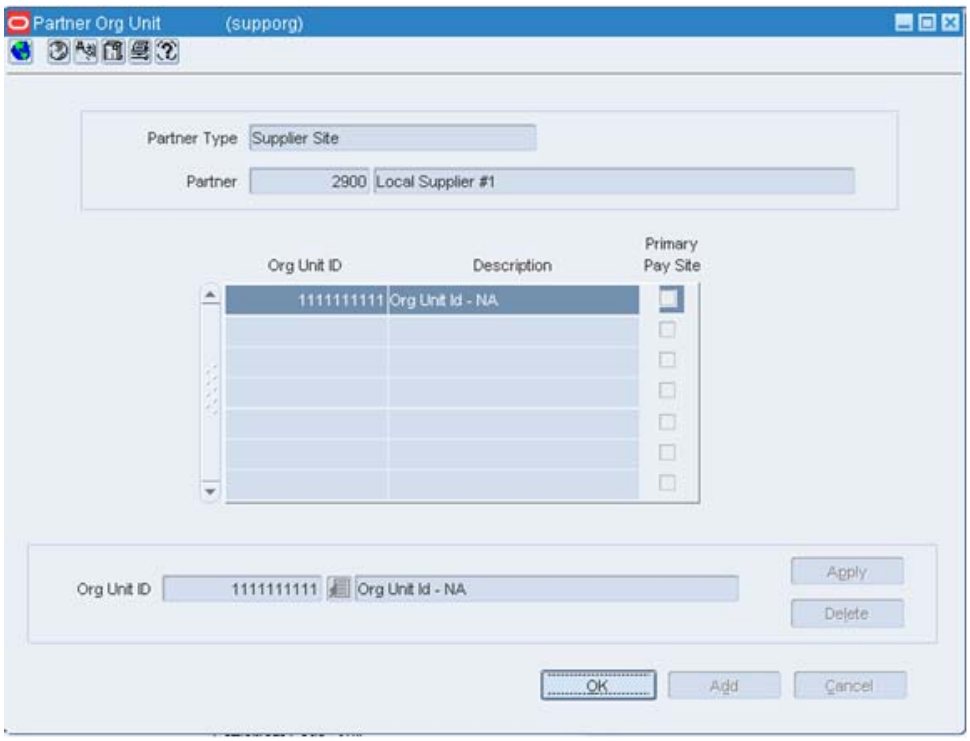
For example, for Consolidation and Operational exchange types, the FIF_CURRENCY_XREF table holds the following entries:

Table 8–1

FIF_EXCHANGE_TYPE	RMS_EXCHANGE_TYPE
C	C
O	O

Supplier Address Types

Within RMS, supplier information (such as Order From and Remit To addresses) is used for generating the purchase orders. PeopleSoft Payables uses supplier information for payment generation. It is important that this information is synchronized.



Suppliers are created in PeopleSoft Payables and exported to RMS. When FINANCIAL_AP is set to A, suppliers cannot be created using the RMS forms. However, after the supplier exists in RMS, all data values for the supplier (except supplier name and status) continue to be updated using the RMS forms. The association of supplier sites to organization units is accessed only in view mode through RMS forms. One supplier site per supplier organization unit combination can be marked as primary payment site.

Where SYSTEM_OPTIONS.FINANCIAL_AP is A, disable auto generate supplier/partner numbers and associated check boxes.

Note: Supplier information is created, updated and inactivated only in PeopleSoft Payables. This information is transferred from PeopleSoft Payables to the participating Oracle Retail applications, where additional retail-specific attributes may be maintained.

Country Codes

When country codes are defined and seeded in RMS, ensure that country codes are mapped to PeopleSoft country codes through RFI DVM mapping. The following is an example of RFI DVM Mapping (Table RFI_XREF_DVM, available in RFI schema in Retail.

Table 8–2

EXT_SYSTEM_ID	COMMON_ID	RETL_ID
USA	700	US
CAN	701	CA

Financial Calendar

The financial calendar within Oracle Retail systems is manually set up and maintained separately from the PeopleSoft General Ledger financial calendar.

Freight Terms

A freight term is an agreement between the retailer and a supplier regarding transportation charges for goods delivered by the supplier. Freight terms are used by RMS as purchase orders are generated.

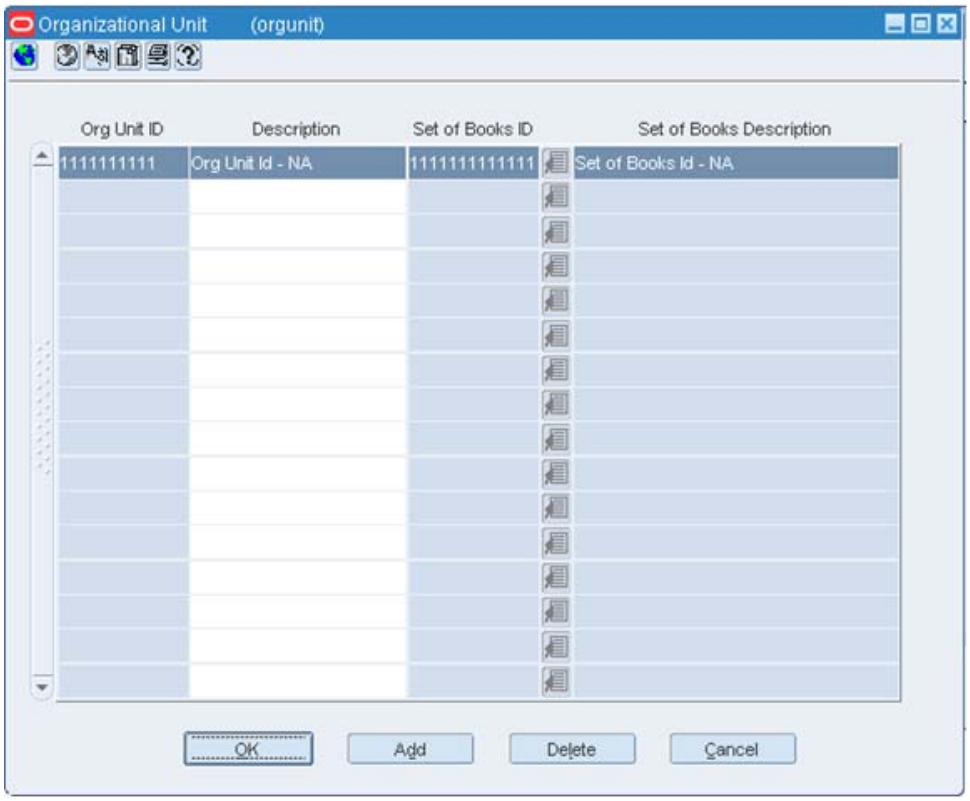
Within the RMS system, freight terms are set up and maintained manually. They also are maintained in PeopleSoft Payables.

Payment Terms and Currency Exchange Rates

Currency exchange rates are created and updated in PeopleSoft General Ledger and exported to RMS. Changes to Retail currency exchange rates are not propagated to PeopleSoft General Ledger. Payment terms set up in PeopleSoft are propagated to RMS but changes to payment terms in RMS are not propagated back to PeopleSoft.

PeopleSoft Financials Units and Site IDs

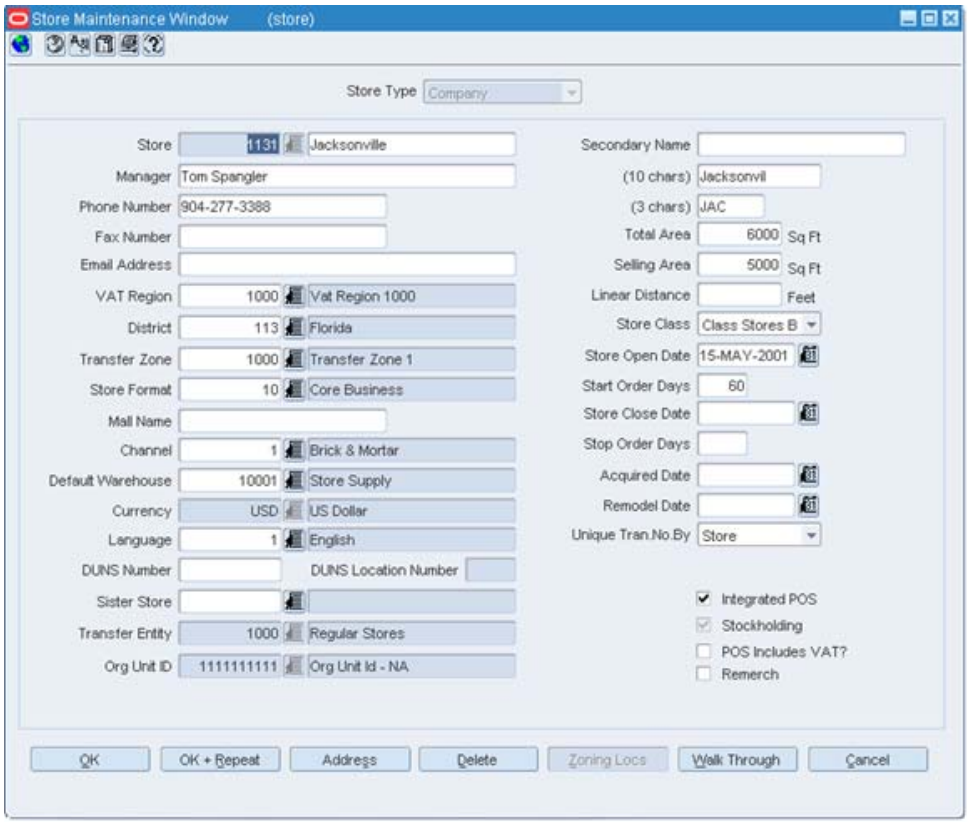
The data concepts of Org Units and Site IDs in RMS mirror the data maintained in PeopleSoft. RMS forms are used to manage and view Oracle Org Units and Site IDs. The RMS windows for Store and Warehouse maintenance allows for the association of each store and warehouse with an Org Unit. The following is an example of the Organizational Unit form:

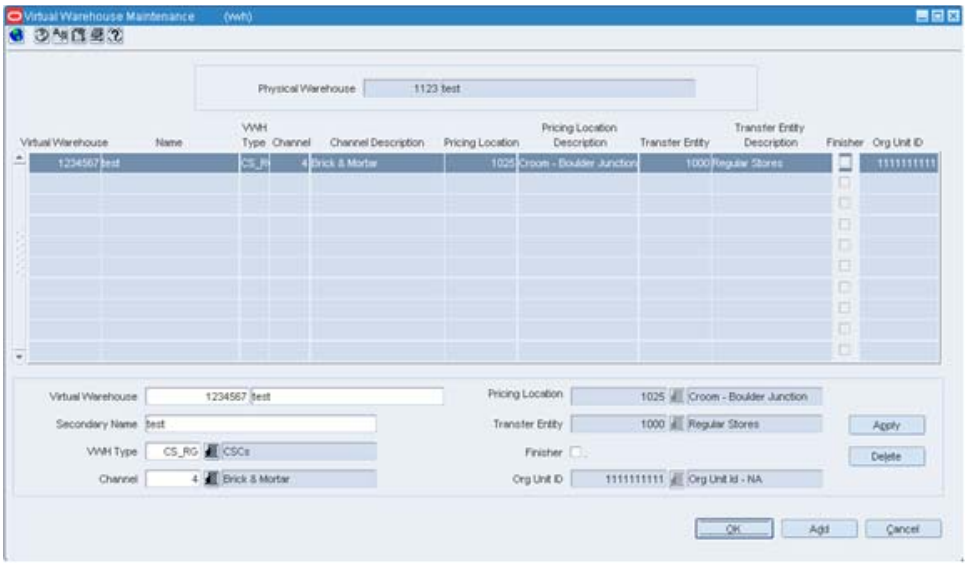


Store and Warehouse Maintenance

The organizational unit is found in the Store Maintenance and Warehouse forms, which allow the PeopleSoft operating unit to be associated with the Store or Warehouse. When RMS is set up for single-channel operation, the organizational unit is set at the physical warehouse level. When RMS is set up for multi-channel operation, the organizational unit is set up at the virtual warehouse level. Financial sales audit and inventory information can then be identified through interface routines and posted to the appropriate general ledger accounts. An organizational unit must be designated for each Store and Warehouse location in the RMS.

The following are examples of the Store Maintenance and Warehouse Maintenance forms:





RMS General Ledger Setup

For RMS and ReSA, manual setup is required for validating the chart of accounts. Valid chart of accounts are created and stored in general ledger cross reference tables. Once the validation is completed, transaction data can be assigned to specific account codes.

Ongoing maintenance of the chart of accounts information (such as adding, changing, or deleting chart of accounts) requires re-validation. In this regard, PeopleSoft General Ledger is the system of record, as it is used to verify the chart of accounts used by Oracle Retail applications. When these applications send a chart of accounts for validation, PeopleSoft General Ledgers issues a message with:

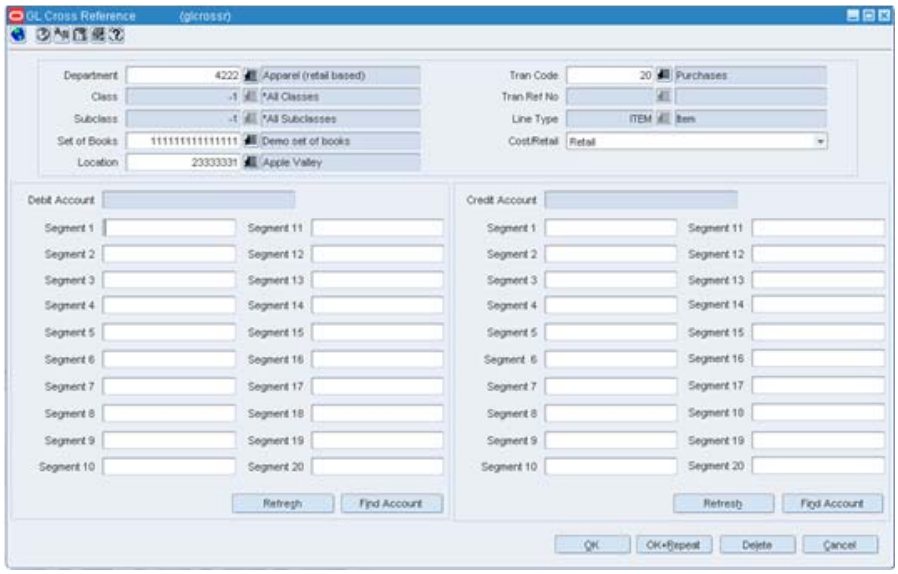
- Valid or invalid status
- Response date
- Chart of accounts

The RMS table, FIF_GL_SETUP, stores the PeopleSoft Set of Books IDs to post financial information. This table must be setup manually after Set of Books IDs are determined. Where system indicator Multiple Set of Books ID is set to N, FIF_GL_SETUP must hold a single Set of Books (SOB) record.

The Set of Books IDs is associated with the chart of accounts when setting up general ledger cross reference.

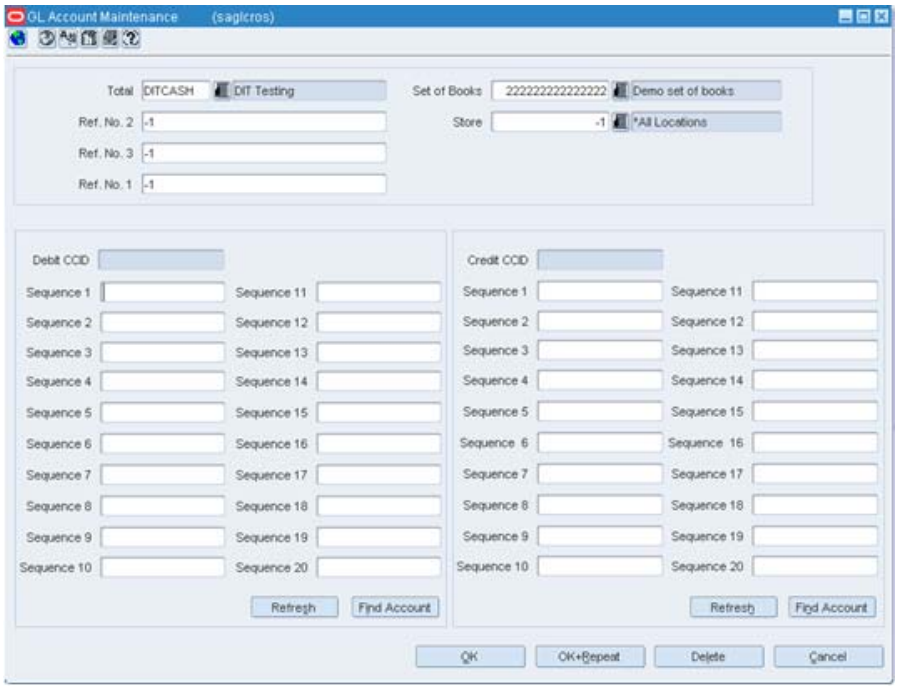
RMS General Ledger Cross Reference

Navigate: RMS Start Menu > Finance> GL Cross Reference. The General Ledger Search window opens. Map Chart of Accounts to department, Class, Subclass, Set Of Books, location, and transaction codes using the GL cross reference form in RMS.



ReSA General Ledger Cross Reference

Navigate: ReSA main menu > Action > Sales Audit > Control > Setup > GL Account Maintenance. The General Ledger Search Form window opens. Where SYSTEM_OPTIONS.FINANCIAL_AP is A, the form requires the entry of valid segment combinations.



Configuring Drill Back and Forward Web Services

Retail web services table, RETAIL_SERVICE_REPORT_URL, must be updated with appropriate URLs to integrate with PeopleSoft Enterprise Financials.

- The records in the table for Services (indicated by RS_TYPE=S) for Account Validation (RAV) and Drill Forward (RDF), must be updated with the URL information from AIA where the services are hosted.

  Note: If Web services are secure, then the SYS_ACCOUNT column must be populated with authentication information in the form of *user name/password*.

- The records in the table for Reports (indicated by RS_TYPE=R) for both RMS and ReIM reports , must be updated with the URL information from the BIP Server where the reports are hosted.

ReIM Data Setup and Configuration

This section describes setup considerations for ReIM data.

System Options

As part of the RMS system options setup script, set the following options as indicated:

- FINANCIAL_AP =A

As part of the ReIM system options setup script, DEFAULT_PAY_NOW_TERMS must be updated with the default term ID.

IM_CURRENCY_LOCALE

Because PeopleSoft Enterprise Financials uses only three decimals, the transactions generated by the Oracle Retail ReIM application must not include more than three decimals.

Update im_currency_locale set currency_cost_dec=3.

Prerequisite: The currency_rates table in RMS should be loaded initially by PeopleSoft Enterprise Financials.

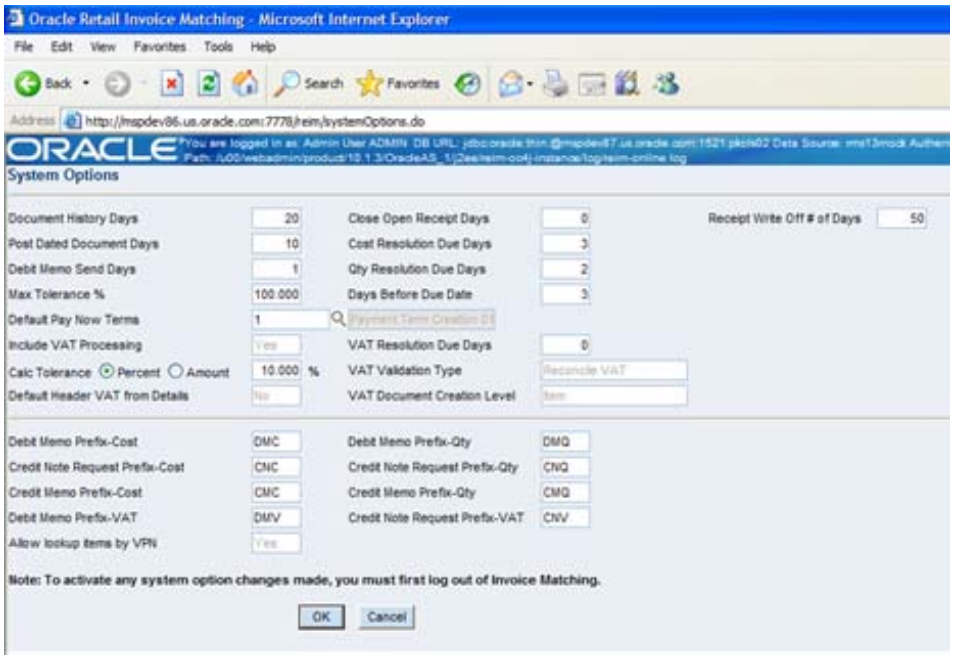
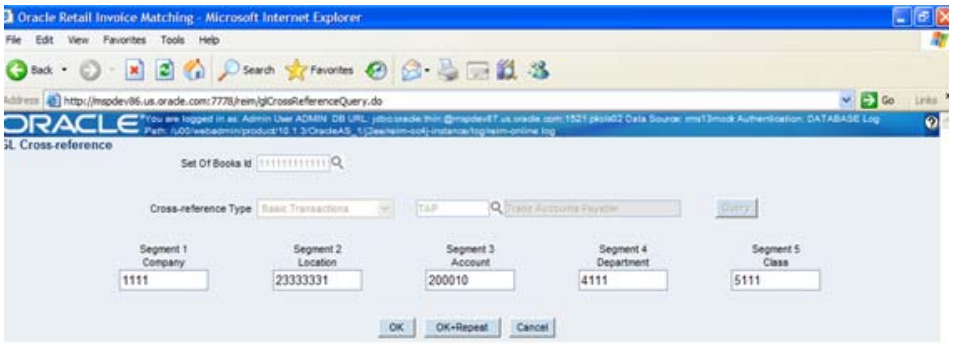


Chart of Accounts Setup

The chart of accounts is set up manually in Oracle Retail applications and in PeopleSoft General Ledger. All account combinations are set up in each Set Of Books. The following is an example of the GL Cross Reference screen:



Note: The Chart of accounts is updated in Oracle Retail applications only after the account is validated through PeopleSoft General Ledger.

Segment Mapping

The retailer determines how many segments are populated. Up to 20 account segments can be specified. The following is an example of how segments are mapped between the ReIM transaction table and PeopleSoft General Ledger:

Table 8–3

ReIM Segments	PeopleSoft General Ledger Chart of Accounts
Segment 1	ACCOUNT

Table 8–3 (Cont.)

ReIM Segments	PeopleSoft General Ledger Chart of Accounts
Segment 2	ALTACCT
Segment 3	DEPTID
Segment 4	OPERATING_UNIT
Segment 5	PRODUCT
Segment 6	FUND_CODE
Segment 7	CLASS_FLD
Segment 8	PROGRAM_CODE
Segment 9	BUDGET_REF
Segment 10	AFFILIATE
Segment 11	AFFILIATE_INTRA1
Segment 12	AFFILIATE_INTRA2
Segment 13	CHARTFIELD1
Segment 14	CHARTFIELD2
Segment 15	CHARTFIELD3
Segment 16	RESOURCE_TYPE
Segment 17	RESOURCE_CATEGORY
Segment 18	RESOURCE_SUB_CAT
Segment 19	BUSINESS_UNIT_PC
Segment 20	PROJECT_ID

If any one of the values in the 20 segments does not match the PeopleSoft General Ledger, the account combination is considered as invalid. The following error message is issued to the user: "Account combination is invalid in the financial system."

Segments 1 and 2 may be set up as dynamic at the Location level, or Segments 4 and 5 can be dynamic at the Department and Class level respectively. Segments defined as dynamic are allowed to be null for certain types of Basic Transaction or Reason Code cross-reference types. When a segment is null, the segment is assigned dynamically when transactions are posted. (Non-dynamic segments cannot be blank.) Validation applies to the segment combination, not to individual segments.

Note: For Tran code TAP, each segment must have a value regardless of whether the segment is dynamic.

Running the Initial Load from PeopleSoft Financials

The initial load for ReIM is run by PeopleSoft and includes the following information:

- Suppliers
- Currency Rates
- Payment Terms

Note: The view, mv_currency_conversion_rates should be refreshed once the initial loads of currencies from PeopleSoft General Ledger are loaded to ReIM.

integration.properties File Setup

To accommodate integration, the integration.properties file within ReIM must be updated with the appropriate URLs for the account validation and drill forward Web services, as listed below:

Note: Drill forward functionality is applicable for PeopleSoft – RMS and ReIM integration. Drill backward functionality is only applicable for PeopleSoft – RMS integration.

```
#webservice WSDL URL for drill forward
webservice.financial.drill.forward.wsdl=@webservice.drill.forward.wsdl@
 webservice.financial.drill.forward.url.targetnamespace=
webservice.financial.drill.forward.targetsystem=
#webservice WSDL URL for account validation
webservice.financial.account.validation=@webservice.account.validation@

webservice.financial.account.validation.namespace=http://www.oracle.com/retail/fin
/integration/services/GlAccountValidationService/v1

webservice.financial.account.validation.local.code=GlAccountValidationService
#webservice username and password for account validation
webservice.financial.account.validation.username=@webservice.account.validation.us
ername@
webservice.financial.account.validation.password=@webservice.account.validation.pa
ssword
@
```

Reports are created by Business Intelligence Publisher for the following:

The URL for each report must be updated in the table, retail_service_report_url. The following table provides sample URLs.

Reporting

Reports are created by Business Intelligence Publisher for the following:

- Merchandise Invoice
- Non-Merchandise Invoice
- Credit Note
- Credit Memo
- Debit Memo
- Receipt Write-Off

The URL for each report must be updated in the table, retail_service_report_url. The following table provides sample URLs:

Table 8–4

Document Type	Report Name	Sample Report URL
MRCHI	Merchandise invoice document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/invreport.xdo
NMRCHI	Non-Merchandise invoice document Report	http://hostname:portno /xmlpserver_nonsso/Guest/REIM13/Finance/invreport/invreport.xdo
CRDNT	Credit Note document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/crnreport.xdo
CRDMEC	Credit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
CRDMEQ	Credit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEC	Debit Memo cost document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/imemoreport.xdo
DEBMEQ	Debit Memo quantity document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
DEBMEV	Debit Memo Tax document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/memoreport.xdo
RWO	Receipt Write Off document Report	http://hostname:portno/xmlpserver_nonsso/Guest/REIM13/Finance/invreport/rworeport.xdo

ReIM Transactional Maintenance

Integration to PeopleSoft General Ledger includes a number of transactions, as described below.

Calculation of TRANS_AMOUNT

The TRANS_AMOUNT field in the im_financial_stage table stores the value of the journal entry to be posted to PeopleSoft General Ledger. (The currency for the calculated amount is the currency assigned to the transaction.) The TRANS_AMOUNT value is calculated as follows:

Table 8–5

Row Description	DEBIT_CREDIT_IND	TRANS_AMOUNT Value
Normal	Debit	Transaction Amount
Normal	Credit	(-1) * Transaction Amount
VAT	Debit	Transaction Amount * Tax Rate
VAT	Credit	(-1) * Transaction Amount * Tax Rate

Note: Transaction Amount is taken from the database column, IM_FINANCIALS_STAGE.AMOUNT.

Generation of Outgoing Data

A staging table accommodates the outgoing transfer of data. The reference key assigned to each document or receipt is used to find data on this table.

Outgoing Data

Table 8–6

From	To	Transactions
ReIM	PeopleSoft Accounts Payable	Invoices Debit Memos Credit Memos Credit Notes
ReIM	PeopleSoft General Ledger	General Ledger accounting entries resulting from the Invoice Matching process, including: Pre-paid invoices Receipt Write-offs
RMS	PeopleSoft General Ledger	Accounting entry data (potentially very high volume)
ReSA	PeopleSoft General Ledger	Accounting entry data (potentially very high volume)

Validation of Accounts When Posting Financial Entries

Valid accounts are stored in the ReIM table, IM_VALID_ACCOUNTS, which includes the Set of Books ID (sob_id) and 20 segments. An ORFI Web service validates accounts against the PeopleSoft Enterprise Financials system. Valid accounts are posted to IM_VALID_ACCOUNTS; invalid accounts are posted to IM_POSTING_DOC_ERROR. The following steps describe the validation process:

1. The ReIM system invokes the account validation Web service to validate the account. (A URL for the ORFI Web service is configured in the integration.properties file.)
2. The posting batch job checks the accounts to be posted against the IM_VALID_ACCOUNTS table.
3. If the account entries are in the table, the transaction is posted to the G/L or AP tables.
4. If the account does not exist in the table, a collection of accounts is built. These collected accounts are validated against the PeopleSoft Enterprise Financials system, and a status is returned.
   - If the status of the collected accounts is valid, the accounts are inserted in the IM_VALID_ACCOUNTS table, and the transactions are posted to the staging tables.
   - If the status of the accounts is NOT valid, the entire collection is flagged as errors, and transactions are postedto IM_POSTING_DOC_ERROR.

Note: ReIM completes the first level of account validation and posts the transaction to staging tables. It is assumed the second level of account validation is done at the end of the extraction process (where transactions are moved from ReIM staging tables to PeopleSoft). If account validation fails at this point, Oracle Data Integrator (ODI) or PeopleSoft must change the account information before transactions are loaded to PeopleSoft, and the account change must be communicated to ReIM .

Maintenance of Valid Accounts

As account information is changed in the PeopleSoft system, the same changes are communicated to, and manually completed in, the ReIM system. After ReIM is updated accordingly, the AccountPurge Batch is run to clear the valid accounts maintained locally in ReIM.

The AccountPurge Batch can clear all valid accounts in the IM_VALID_ACCOUNTS table or only those that are considered updates in PeopleSoft.

Usage:

```
AccountPurge batch-alias-name  PURGE [ALL | <Accounts>]
```

Where

The first argument is batch alias name.

The second argument is the word PURGE.

The third argument is either ALL or specific accounts to be deleted from the local table.

Building and Posting Reference IDs

Drill back and drill forward functionality uses Reference ID to locate documents and receipts. A Reference ID is a combination of document type and document (or receipt) ID, as illustrated in the table below:

Table 8–7

Type	Doc ID	Receipt ID	Reference ID
Merchandise Invoice	101	Null	MRCHI#101
Non-Merchandise Invoice	102	Null	NMRCHI#102
Receipt	Null	103	RECEIPT#103

For documents, the Resolution Posting Batch program builds the Reference ID using the standard, Document Type + DeLimiter + Doc_id. For receipts, the program builds the Reference ID using the standard, Document Type + DeLimiter + Receipt_id.

To enable drill down functionality, Reference IDs are loaded to staging tables. FinancialsAPStageDao and FinancialsGLStageDao are populated, as are IM_RWO_SHIPMENT_HIST and IM_RWO_SHIPSKU_HIST.

Drilling Back

Drilling back allows users to view the source of posted PeopleSoft transactions that originated in Oracle Retail systems (from a voucher to an invoice, for example).

When drilling back from PeopleSoft Enterprise Financials, users are not directed to an actual screen within RMS, ReIM or ReSA. Rather, a retail Web service generates and launches a URL to a BI Publisher report. The report contains the information that typically appears on the appropriate retail screen.

Depending on the information requested by the user, PeopleSoft invokes web service in the ORFI layer which in turn calls the report locator service available in RMS. This service returns back BIP report reference URL which is passed back to PeopleSoft and report gets launched in browser window.

Information from the reference key determines what kind of report URL to issue. For example, if the retail key has a prefix of ReIM, the ReIM_REPORT_URL function is called, else the RMS_REPORT_URL function is called to retrieve the appropriate report URL. If the key does not match any key in the retail systems, an error message is launched.

Drilling Back to RMS and ReSA from PeopleSoft Enterprise Financials

The following function determines which RMS report to return to the user:

```
RMS_REPORT_URL() -
     O_error_message     IN OUT     RTK_ERRORS.RTK_TEXT%TYPE
     O_rpt_url           IN OUT     RETAIL_SERVICE_REPORT_URL.URL%TYPE
     I_ref_key           IN         KEY_MAP_GL.REFERENCE_TRACE_ID%TYPE
```

The appropriate report URL is found and issued as follows:

1. The ref_trace_type is found on KEY_MAP_GL by matching I_ref_key with the KEY_MAP_GL.REFERENCE_TRACE_ID column.
2. When ref type is determined, the re_trace_type is used to find the appropriate report URL on the RETAIL_SERVICE_REPORT_URL table.
3. The value of I_ref_key is appended to the end of the URL retrieved from the table.
4. The URL is sent back to the calling function.
5. If I_ref_key does not exist on KEY_MAP_GL, an error message is sent back to the calling function.

Drilling Back to ReIM from PeopleSoft Enterprise Financials

The following drill back options are available for viewing information within the ReIM system:

- Using Document ID, users can drill back to ReIM to view information related to a voucher or payment. The report includes information from im_doc_head and im_invoice_detail, the same data shown on the Document Maintenance Header screen within ReIM.
- Using the Receipt ID, users can drill back to view information from the Receipt Write-off History screen. Receipt write-offs occur either when an open receipt is closed in ReIM or if a receipt is purged in RMS before it is fully matched. Details come from the IM_RWO_SHIPMENT_HIST and IM_RWO_SHIPSKU_HIST tables.

The function below determines which of the two ReIM reports to return to the user:

```
REIM_REPORT_URL() -
     O_error_message     IN OUT     RTK_ERRORS.RTK_TEXT%TYPE
     O_rpt_url           IN OUT     RETAIL_SERVICE_REPORT_URL.URL%TYPE
     I_ref_key           IN         KEY_MAP_GL.REFERENCE_TRACE_ID%TYPE
```

The I_ref_key contains the reference ID, which ultimately determines the type of report required. The appropriate BI Publisher report URL is found on the RETAIL_SERVICE_REPORT_URL table.

In general, if the reference ID has a prefix of RECEIPT, the report type (RS_CODE) is RCPT. Otherwise, the report type is DOC. For example:

Table 8–8

Reference ID	Report Type (RS_CODE)
MRCHI#101	DOC
NMRCHI#102	DOC
RECEIPT#103	RCPT

The following is an example of a BI Publisher URL that is generated upon drilling back to PeopleSoft Enterprise Financials for information on an invoice in ReIM, using Document ID as the search parameter:

```
http://mspdev6970vip:7777/BIPublisher/Guest/ReIM/13.0.3/doc/tsf_det.xdo ?doc_
id=101
```

Where

- http://mspdev6970vip:7777/BIPublisher = the BI Publisher application server address and port
- Guest/ReIM/13.0.3 = the directory/folder location
- doc/tsf_det.xdo ? = report name (Document Report)
- doc_id=101 = the parameter name and value (Document ID 101)

The following is an example of an Oracle Business Intelligence Publisher URL that is generated upon drilling back to PeopleSoft Enterprise Financials for information on an invoice in ReIM, using Receipt ID as the search parameter:

```
http://mspdev6970vip:7777/BIPublisher/Guest/ReIM/13.0.3/doc/tsf_det.xdo ?receipt_
id=101
```

Where

- http://mspdev6970vip:7777/BIPublisher = the BI Publisher application server address and port
- Guest/ReIM/13.0.3 = the directory/folder location
- doc/tsf_det.xdo ? = report name (Receipt Report)
- receipt_id=101 = the parameter name and value (Receipt ID 101)

Drilling Forward

Drilling forward allows users to see detailed information about retail transactions that have been posted to PeopleSoft Enterprise Financials. When drilling forward, users are directed to selected "view-only" screens.

Drilling Forward From RMS/ReSA to PeopleSoft Enterprise Financials

The following forms may be used to drill forward from RMS/ReSA:

- RMS StartMenu->Finance->Transaction Data View (trandata.fmb)

- RMS StartMenu->Ordering->Fixed Deals->Fixed Deal Transaction Data View (fdltrandata.fmb)
- RMS StartMenu->Action->Sales Audit->Sales Audit Transaction Data View (satrandata.fmb)

Drilling Forward From ReIM to PeopleSoft Enterprise Financials

For drilling forward, the ORFI Web service uses the Invoice ID and Accounting Entry parameters. The ReIM system uses these parameters together as the Reference ID.

- From the Document Maintenance screen, users can drill forward to PeopleSoft Enterprise Financials accounts payable to view voucher and payment status. The information is displayed on a read-only Payment Doc Status Inquiry screen. Drill forward access to the accounts payable system is available only for pre-paid invoices (but not for manually pre-paid invoices).

  To drill down to the payables/ledger screens, the user invokes the Web service as follows:

  ```
  Invoice ID/Accounting Entry parameter = Reference ID
  ```

- From the Receipt Write-off History screen in ReIM, users can drill forward to the PeopleSoft G/L system to view the status of related journal entries. Drill forward access to the G/L system is available only for pre-paid and manually pre-paid invoices.

Note: For more information on drilling forward, see the *Oracle Retail Financial Integration for Oracle Retail Merchandise Operations Management and Oracle Financials Suite (E-Business Suite Financials or PeopleSoft Financials)*.

9

Batch Processes

This chapter provides the following:

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

Batch Architectural Overview

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

Services retrieve the data on which the batch processes work to complete their tasks. As noted in Chapter 3, "Technical Architecture," the service layer consists of a collection of Java classes that implements business logic (data retrieval, updates, deletions, and so on) 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.
- Although ReIM runs at all times, batch processes should be executed when the fewest number of users are in the system, such as at night.

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.

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
- Discrepancy purge
- Disputed credit memo action rollup
- Reason credit action rollup
- User Maintenance

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. They include the following:

- Financial posting
- Receiver adjustment

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 9–1

Batch Name	Class (oracle.retail.reim.batch.jobs)
Tables purge	TablesPurgeBatch
Account purge	AccountWorkspacePurgeBatch
Discrepancy purge	DiscrepancyPurgeBatch
EDI Injector	EdiInjectorBatch
Auto-match	AutoMatchBatch
Receipt write-off	ReceiptWriteOffBatch
Reason code action rollup	ReasonCodeActionRollupBatch
Disputed credit memo action rollup	DisputedCreditMemoResolutionRollupBatch
Financial posting	FinancialPostingBatch
EDI Invoice download	EdiDownloadBatch
Complex deal upload	ComplexDealUploadBatch
Fixed deal upload	FixedDealUploadBatch
User Maintenance	UserMaintenanceBatch

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 9–2

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.	
Discrepancy purge	The discrepancy purging program deletes data from database tables while maintaining database integrity. This program deletes records from ReIM that have discrepancies of zero.	
EDI 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. The processing consists of three levels - summary, detail, and header (Tax only).	▪ EDI Injector ▪ Receipt upload (Merchandising system, such as RMS)

Table 9–2 (Cont.)

Batch Processes	Details	Batch Dependencies
Receipt write-off	In order for retailers to track received goods not invoiced, they must have the ability to 'write-off' these goods for financial tracking. ReIM has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the Receipt write-off process is run, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, the receipt is 'written-off,' and the invoice match status is closed.	Auto-match and any associated processing must run prior to this batch processing.
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.	
Disputed credit memo action rollup	The disputed credit memo action rollup process checks the records on the IM_REVERSAL_RESOLUTION_ACTION table and rolls up the credit memo detail lines by document/item/reason code. The rollup occurs only if all lines on a disputed credit memo have been completely resolved (that is, no cost or quantity discrepancy records remain for the credit memo). After the rollup, a new set of detail lines associated with the resolution reason codes replace the original set of detail lines associated with the debit reason codes on the IM_DOC_DETAIL_ REASON_CODES table.	The disputed credit memo action rollup must occur before resolution posting and after receiver adjustment.

Table 9–2 (Cont.)

Batch Processes	Details	Batch Dependencies
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. For each invoice, the batch process writes applicable financial accounting transactions to either of the following tables: IM_FINANCIALS_STAGE The AP staging tables, IM_AP_STAGE_HEADER and IM_AP_STAGE_DETAIL, if the RMS System-Options table: FINANCIAL_AP = O.	
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.	The RMS staged data must be purged after the upload
Fixed deal upload	This module reads data from RMS staging tables, creates credit memos, debit memos, and credit note requests out of those, and stores the supporting deal data on a ReIM table for later use during posting.	The RMS staged data must be purged after the upload
User Maintenance	This module synchronizes the business role member data in ReIM with the users in the configured LDAP. It will remove user entries from ReIM tables for which users no longer exist (according to the LDAP entries).	

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
- FAILED_WRAPUP = 3
- UNKNOWN = -1

Batch Log and Error File Paths

Log file locations are determined by the retailer through the logj4.properties file. If an error occurs that causes a batch process to suddenly come to a complete halt, the system writes to the configured log appender. See Chapter 2, "Backend System Administration and Configuration," for more information.

Multi-Threading Batch Processes

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

Complex Deal Upload (ComplexDealUploadBatch)

This process is threaded by a group (or bulk) of deals. Each group 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 can either be run as a single thread or it can be threaded by the location hierarchy.

A Note about Restart and Recovery

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

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

- EDI 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 BatchRunner framework. This framework is responsible for bootstrapping the Spring container and 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 BatchRunner requires the application libraries (JAR files) to be on the classpath in order to execute successfully. Retailers wishing to configure the BatchRunner manually should consult the generic UNIX batch script generated during the install process for assistance in determining which libraries should be included for a particular batch process.

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 [ALL|<table name>] [NOCOMMIT|COMMIT]
```

The first argument is batch alias name. The second argument is the word PURGE. The third argument is either ALL or a single table name. Table name can be any one of the following:

- IM_DOC_GROUP_LIST
- IM_DOC_GROUP_HEAD
- IM_PARENT_INVOICE
- IM_REASON_CODES
- IM_PARTIALLY_MATCHED_RECEIPTS
- IM_TOLERANCE_DEPT_AUDIT
- IM_TOLERANCE_SUPP_AUDIT
- IM_TOLERANCE_SUTRT_AUDIT
- IM_TOLERANCE_SYS_AUDIT

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

SQL Queries

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

Manual Propagation (Cascade) of Deletes to Child Tables

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

Cascade Relationships

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

Step 1

```
Delete from: IM_DETAIL_MATCH_INVC_HISTORY
Delete from: IM_INVOICE_DETAIL_ALLOWANCE
Delete from: IM_QTY_DISCREPANCY_ROLE
Delete from: IM_QTY_DISCREPANCY_RECEIPT
```

Step 2

```
Delete from: IM_DOC_DETAIL_COMMENTS
Delete from: IM_MANUAL_GROUP_INVOICES
Delete from: IM_DOC_HEAD_COMMENTS
Delete from: IM_INVOICE_DETAIL
Delete from: IM_DOC_HEAD_LOCK
Delete from: IM_FINANCIALS_STAGE
Delete from: IM_COST_DISCREPANCY
Delete from: IM_RESOLUTION_ACTION
Delete from: IM_REVERSAL_RESOLUTION_ACTION
Delete from: IM_SUMMARY_MATCH_INVC_HISTORY
Delete from: IM_QTY_DISCREPANCY
Delete from: IM_DOC_DETAIL_REASON_CODES
Delete from: IM_FINANCIALS_STAGE_ERROR
Delete from: IM_DOC_NON_MERCH
Delete from: IM_COMPLEX_DEAL_DETAIL_TAX
Delete from: IM_DOC_DETAIL_RC_TAX
Delete from: IM_DOC_NON_MERCH_TAX
Delete from: IM_FIXED_DEAL_DETAIL_TAX
Delete from: IM_INVOICE_DETAIL_ALLW_TAX
Delete from: IM_INVOICE_DETAIL_TAX
Delete from: IM_ITEM_TAX_AUDIT
Delete from: IM_ORDER_ITEM_TAX_AUDIT
Delete from: IM_TAX_DISCREPANCY
Delete from: IM_DOC_TAX
```

Step 3

```
Delete from: IM_DOC_HEAD
```

Cascade relationships are wired in the TablesPurge.java.

Assumptions and Scheduling Notes

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

Major Modules

TablesPurgeBatch

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

Primary Tables Involved

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

- IM_DOC_GROUP_LIST
- IM_DOC_HEAD
- IM_PARENT_HEAD
- IM_REASON_CODES

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

AccountWorkspacePurge

Major Tables

IM_VALID_ACCOUNTS

Discrepancy Purge Batch Design

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

Usage

The following arguments are applicable for the DiscrepancyPurgeBatch process:

```
reimdiscrepancypurge batch-alias-name PURGE [ALL|<table name>] [NOCOMMIT|COMMIT]
```

Where the first argument is batch alias name. The second argument is the word PURGE. The third argument is either ALL or a single table name. Table name can be any one of the following:

- IM_COST_DISCREPANCY
- IM_QTY_DISCREPANCY

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

SQL Queries

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

Major Modules

DiscrepancyPurge

Major Tables

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

EDI Invoice 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_xxx, IM_PARENT_xxx tables.

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

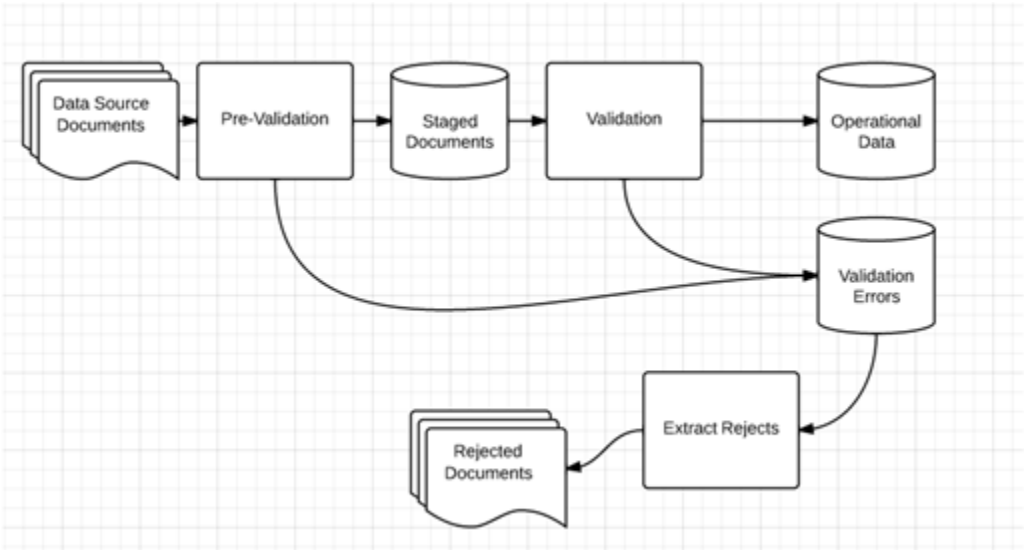
This process can be run ad-hoc but in general it should be run before the auto-match process.

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
- IM_PARENT_INVOICE
- IM_PARENT_INVOICE_TAX
- IM_PARENT_INVOICE_DETAIL
- IM_PARENT_NON_MERCH
- IM_PARENT_NON_MERCH_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 three inputs into the auto-match process include the following:

- Invoices
- Receipts
- Purchase orders

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

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

The auto-match process attempts to match the invoices to receipts to the best of its abilities. The process 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.

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

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

This process identifies any cost discrepancies prior to the arrival of receipts. If no receipts exist for the PO location, the invoices are sent to the cost pre-matching algorithm. Cost pre-matching is where unit costs on the invoice are compared with unit costs on the purchase order at a line level. If a match can be obtained, the invoice remains in ready-for-match status and is retrieved again for matching once the receipt comes in. If no match can be obtained, a cost discrepancy is created and routed immediately.

Summary matching

Invoices are grouped with receipts based upon purchase order location. A match is attempted for all invoices and receipts for the PO location. The invoices' total extended costs are summed and compared with the receipts' total extended costs. Based on a supplier option, the invoices' total quantity is summed and compared with the receipts' summed total quantity. If a match is achieved, all invoices and receipts are set to matched status. Otherwise, one-to-one matching is attempted for the PO location.

One-to-one invoice matching

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

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

Detail matching

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

Header matching

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

Assumptions and Scheduling Notes

Consider the following assumptions and scheduling notes.

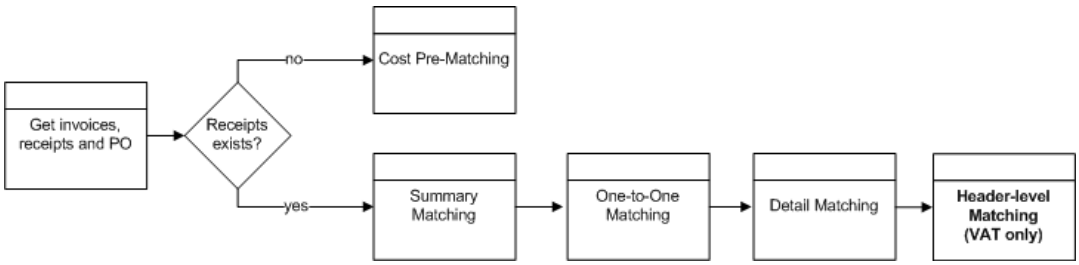
- Although not recommended, auto-match can be run during the day when there are users online interacting with the system.
- Both the invoice unit cost and the unit cost of the PO must be expressed in the same currency. In order to compare the invoice unit costs with the PO's unit costs, auto-match does not engage in currency conversion. The system assumes that tolerance costs are always in the system's primary currency. If RMS is the applicable merchandising system, auto-match performs currency conversion if the currency on the order is different from the primary currency. RMS existing currency conversion engine is used to perform this conversion. If RMS is not being utilized, another currency conversion engine must be provided to support this functionality.
- The quantities on the invoice must be expressed in the same unit of measure as the quantities on the receipt. Auto-match performs no unit of measure conversion.
- The batch process runs after EDI Injector (Invoice Matching) and Receipt upload (Merchandising system, such as RMS).
- Supplier options. All suppliers must have options defined in order for their invoices to be processed by the system, and the terms defined for those suppliers have to be completely updated in RMS. In order to support the use of suppliers in ReIM, the ENABLED_FLAG (set to Y), START_DATE_ACTIVE and END_DATE_ACTIVE are the required entries in the TERMS_DETAIL table in RMS.

Post Processing

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

High-Level Flow Diagram

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



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_DEPT
- IM_TOLERANCE_SUPP
- IM_TOLERANCE_SYSTEM
- IM_COST_DISCREPANCY
- IM_QTY_DISCREPANCY
- IM_QTY_DISCREPANCY_RECEIPT
- IM_QTY_DISCREPANCY_ROLE
- IM_SUPPLIER_OPTIONS
- IM_SYSTEM_OPTIONS

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 "Credit Note Auto-Matching" in Chapter 4.

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. For information, see "Credit Note AutoMatch Batch Multi-threading Options" in Chapter 2.

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:

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

To enable the use of all three keys, the customizable reference fields in the credit notes and credit note requests must be populated. For information, see "Credit Note Auto-Matching" in Chapter 4. The customizable Ref No. 3 field holds the credit note request ID, and Ref No. 4 field holds the original invoice ID. In case none of the customizable 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 in a document-key set when no match is found:

1. Credit Note Request ID (configurable key)
   - Summary Matching (matching algorithm)
   - One to One Matching (matching algorithm)
   - Line-level Matching (matching algorithm)
2. Original Invoice ID (configurable key)
   - Summary Matching (matching algorithm)
   - One to One Matching (matching algorithm)
   - Line-level Matching (matching algorithm)
3. PO/Location (configurable key)
   - Summary Matching (matching algorithm)

- One to One Matching (matching algorithm)
- 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 the department, and lastly with respect to the system. For information, see "Credit Note Auto-Matching" in Chapter 4.

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. For information, see "Credit Note AutoMatch Workspace Cleanup Setting" in Chapter 2.

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

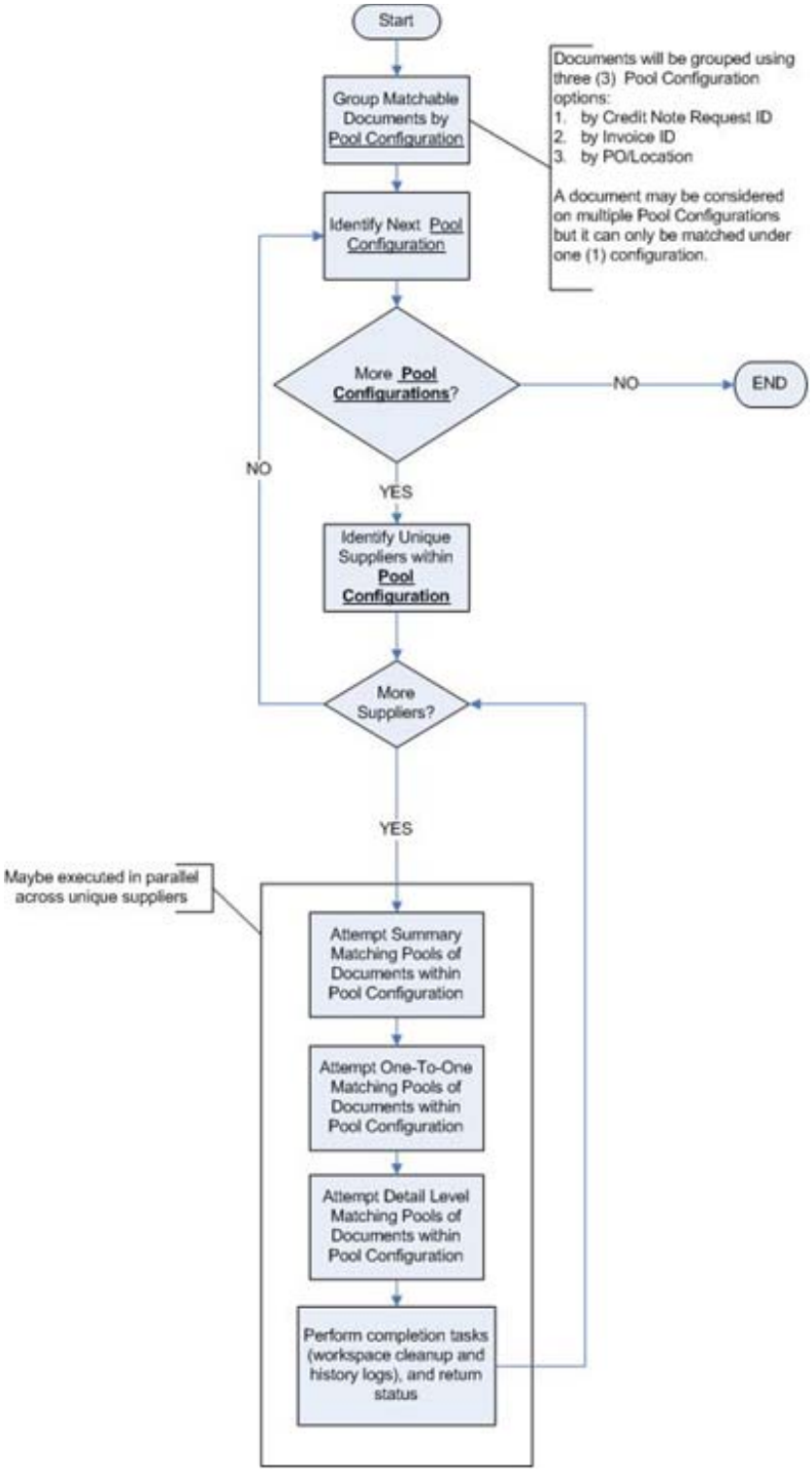
Consider the following assumptions and scheduling notes.

- Both the credit note and credit note request unit cost must be expressed in the same currency. If the currency on the credit note and credit request is the same, but differs from the primary system currency, then an attempt will be made to perform currency conversion only if RMS is the applicable merchandising system.
- 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 9–3

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_SUPP	Tolerance properties associated with a supplier. The data is required when performing matches within tolerances at the supplier level.
IM_TOLERANCE_DEPT	Tolerance properties associated with a department. The data is required when performing matches within tolerances at the department level.
IM_TOLERANCE_SYSTEM	Tolerance properties associated with a system. The data is required when performing matches within tolerances at the system level.
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 9–4

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 tables are populated for compatibility with the existing Invoice Matching discrepancy related data model.

Table 9–5

Table Name	Contents
IM_QTY_DISCREPANCY	Quantity discrepancy records.
IM_QTY_DISCREPANCY_ ROLE	The Associate roles with the access to generate discrepancies.
IMP_QTY_ DISCREPANCY_CNR	Quantity discrepancies on the credit note get associated with participating credit note requests.
IM_COST_DISCREPANCY	Cost discrepancy records.
IM_COST_ DISCREPANCY_CNR	Cost discrepancies on the credit note get associated with participating credit note requests.

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 9–6

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.
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 9–7

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

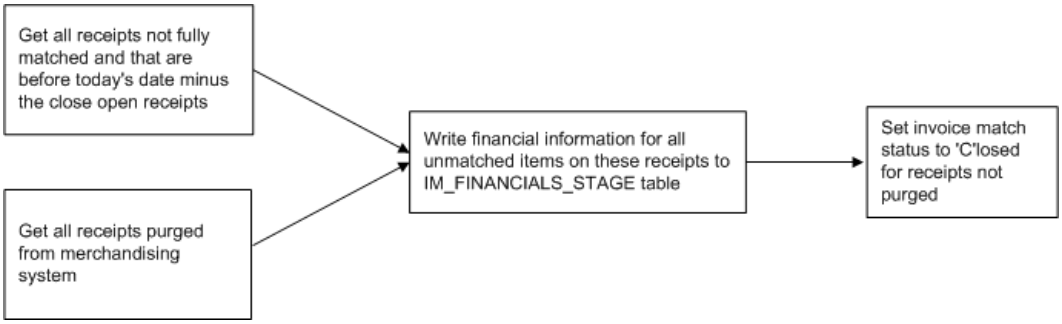
The following arguments are applicable for the Receipt Write-Off Batch process:

```
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.

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

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

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

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

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

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

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

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

Usage

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

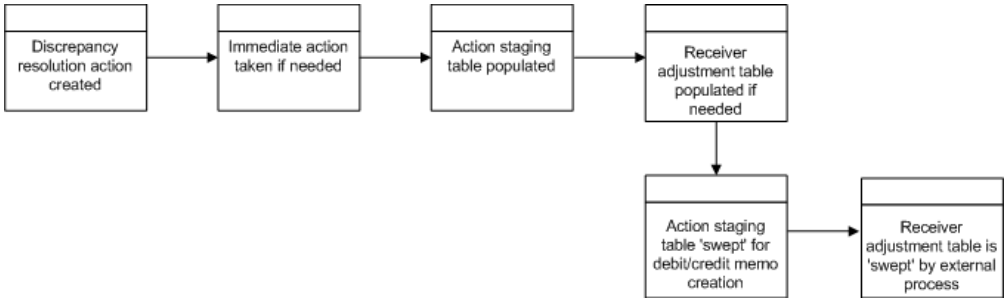
```
reimrollup atch-alias-name
```

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

Disputed Credit Memo Action Rollup Batch Design

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

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

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

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

Assumptions and Scheduling Notes

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

Primary Tables Involved

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

- IM_DOC_HEAD. This table holds the document header information.
- IM_DOC_DETAIL_REASON_CODES. This table holds the document detail information by item/reason code. Before resolution rollup, this table holds the document detail information based on the original debit reason codes. After

resolution rollup, this table holds the document detail information based on the reason codes used to resolve the disputed credit memo lines.

- IM_REVERSAL_RESOLUTION_ACTION. This table holds the resolution actions the user takes to approve or deny the disputed credit memo line.
- IM_COST_DISCREPANCY. This table holds the disputed credit memo lines for a debit memo cost reversal.
- IM_QTY_DISCREPANCY. This table holds the disputed credit memo lines for a debit memo quantity reversal.
- IM_QTY_DISCREPANCY_ROLE. This table holds the routing information for a credit memo quantity.

Financial Posting Batch Design

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

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

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 system.properties.
- 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 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 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.

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
```

Assumptions and Scheduling Notes

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

Primary Tables Involved

Note: For descriptions of RMS tables, see the *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
```

Assumptions and Scheduling Notes

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

Primary Tables Involved

Note: For descriptions of RMS tables, see the *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.

User Maintenance Batch Design

The user maintenance program deletes data from database tables while maintaining database integrity. This program deletes user based records from ReIM that are not in sync with the configured LDAP. Main features of the process are as follows:

Usage

The following arguments are applicable for the UserMaintenanceBatch process:

```
reimusermaintenance batch-alias-name
```

Algorithms

The UserMaintenance batch process includes the following algorithm.

- User Synchronization

  Tables with records that hold a username identifier will be verified against the configured LDAP. If a user does not exist in the LDAP, then the user mapping will be purged.

Primary Tables Involved

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

- IM_BUSINESS_ROLE_MEMBER