Oracle® Retail Invoice Matching

Operations Guide

Release 12.0.10

April 2009

ORACLE

Oracle® Retail Invoice Matching Operations Guide, Release 12.0.10

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

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

(i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server – Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.

 (ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.

(iii) the **SeeBeyond** component developed and licensed by Sun MicroSystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.

(iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.

(vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.

(vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.

(viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

(ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

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

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

Contents

Preface

Oracle Retail Operations Guides are designed so that you can view and understand the applications behind-the-scenes processing, including such information as the following:

- Key system administration configuration settings
- Technical architecture
- Functional integration data flow across the enterprise
- Batch processing

Audience

Anyone who has an interest in better understanding the inner workings of the RPM system can find valuable information in this guide. There are three audiences in general for whom this guide is written:

- Systems analysts and system operation personnel who need information about internal Oracle Retail Invoice Matching processes.
- Integrators and implementers who are responsible for implementing Oracle Retail Invoice Matching.
- Business analysts who need information about Oracle Retail Invoice Matching processes and interfaces.

Related Documents

For more information, see the following documents in the Oracle Retail Invoice Matching Release 12.0.10 documentation set:

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

Customer Support

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

https://metalink.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

Conventions

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

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

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

A hyperlink appears like this.

Third-Party Open-Source Applications in ReIM 12.0.10

Oracle Retail Invoice Matching includes the following third-party open-source applications:

Software Provider: Apache Software Foundation
Software Name: Apache Struts
Software Version: 1.1
Jar File Name: struts-1.1.jar
Provider Web Site: http://jakarta.apache.org/

Software Provider: Apache Software Foundation
Software Name: org/apache/log4j/
Software Version: 1.2.14
Jar File Name: log4j-1.2.14.jar
Provider Web Site: http://logging.apache.org/log4j/docs/

Software Provider: IBM
Software Name: ICU Project
Software Version: n/a
Jar File Name: icu4j.jar
Provider Web Site: http://www.icu-project.org/

Software Provider: IBM
Software Name: IBM Alpha Works
Software Version: n/a
Jar File Name: decimal.jar
Provider Web Site: http://www.ibm.com

Software Provider: Apache Software Foundation
Software Name: org.apache.commons.logging
Software Version: 1.0.3
Jar File Name: commons-logging-1.0.3.jar
Provider Web Site: http://jakarta.apache.org/commons/

Software Provider: Apache Software Foundation
Software Name: Jakarta Commons Lang
Software Version: 2.0
Jar File Name: commons-lang.jar
Provider Web Site: http://jakarta.apache.org/commons/

Software Provider: Apache Software Foundation
Software Name: Jakarta Commons Digester
Software Version: 1.5
Jar File Name: commons-digester-1.5.jar
Provider Web Site: http://jakarta.apache.org/commons/

Software Provider: Apache Software Foundation
Software Name: org.apache.commons.collections
Software Version: 3.2
Jar File Name: commons-collections-3.2.jar
Provider Web Site: http://jakarta.apache.org/commons/

Software Provider: Apache Software Foundation
Software Name: org.apache.commons.beanutils
Software Version: 1.6
Jar File Name: commons-beanutils-1.6.jar
Provider Web Site: http://jakarta.apache.org/commons/

Software Provider: Apache Software Foundation
Software Name: i18n Tag library
Software Version: 1.1
Jar File Name: i18n.jar
Provider Web Site: http://jakarta.apache.org/taglibs/doc/i18n-doc/intro.html

Software Provider: iText
Software Name: iText
Software Version: 1.1
Jar File Name: iText.jar
Provider Web Site: http://lowagie.com/iText/ or
http://sourceforge.net/projects/iText/

Software Provider: Spring
Software Name: Spring Framework
Software Version: 2.5
Jar File Name: spring-2.5.jar
Provider Web Site: http://springframework.org

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 chargeback 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.

A Note About 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 only able to communicate 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.

Hardware and Software Requirements

See the Oracle Retail Invoice Matching Installation Guide for information about the following:

- Hardware and software requirements
- Oracle Retail application software compatibility

System Assumptions

- 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.

- VAT

  If VAT is turned on, the retailer must have VAT regions, VAT items, and VAT codes set up in the merchandising system (such as RMS) to support validation of invoiced VAT 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.

  - VAT_IND is set to Y.
  - VAT_VALIDATION_TYPE is set to Reconcile VAT, Always Use Invoice VAT, or Always use System VAT.
  - The DEFAULT_VAT_HEADER is set to Y or N.
  - VAT_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. The key system parameters contained in this file are described in this section.

In the properties file, certain values are preceded by a # sign. This indicates the line is a comment and is not used as a setting

Every setting in the reim.properties file is configurable. When retailers implement code in their environment, they must update these values to their specific settings, taking system performance, for example, into consideration.

See the section, "Internationalization and Localization", later in this chapter for additional descriptions of reim.properties values.

Connection Information for the Database

This portion of the file identifies what JDBC driver the system is utilizing. (For more information about JDBC, see Chapter 3 – "Technical Architecture".) This data also includes what datasource (merchandising system) that ReIM is utilizing for its foundation data and the environment information associated to that datasource. The following settings—with example values-- apply:

```
# Datasource configuration
datasource.url=jdbc:oracle:thin:@mspvip72.us.oracle.com:1521:dvolr021
datasource.username=rmssedev121user
datasource.password=password
datasource.schema.owner=rmssedev121
datasource.bean.driver=com.retek.reim.foundation.rms12
datasource.datasource=rms12
```

Authentication Section

Authentication within ReIM addresses the legitimacy and the security privileges of users, an important aspect of the system's security handling process.

Within the authentication section of this file, the retailer selects either LDAP or DATABASE, depending upon which is applicable.

For example:

```
authentication_source=@deploy.authentication.source@
IConnectionSettingsDAO=com.retek.reim.merch.utils.PropertyFileLdapSettingsDao
ISecurityDao=com.retek.reim.merch.utils.ReIMLdapSecurityDao
ISecurityRelationshipDAO=com.retek.reim.merch.utils.LDAPSecurityRelationshipDAO
```

For retailers selecting LDAP, see Chapter 5 – "Interfaces and File Layouts" and see the ReIM Installation Guide for more settings and information.

Minimum and Maximum Pool Size

The pool size pertains to the number of available database connections that the retailer intends to keep available in the pool. A system administrator is encouraged to adjust these values per configuration to match the retailer's anticipated number of users. The default values are intended to be a mere starting point.

In the example below, a minimum of five connections are available, and no more than 10 are available.

```
# Minimum and maximum pool size to maintain
pool.min=5
pool.max=10
```

Pool SQL Trace

When this setting is true, SQL statements are shown. This setting might be used during development, build time, performance tuning, debugging, troubleshooting, and so on.

In the example below, the retailer has decided *not* to log the SQL statements.

```
pool.sqltrace=FALSE
```

Pool Implicit Cache

This parameter specifies whether or not the application needs to implicitly inform the connection pool cache that a connection is being used.

Security Ports

Within the enterprise, a port is an endpoint to a logical connection and the way a client program specifies a specific server program on a computer in the network. A security port is analogous to an address for a given machine. The security port "listens" at this address, and if the system needs to process security-related data, it must "talk" only to that address to do so. The security-related processing can only occur at the place where the listening is occurring.

```
# Security Ports
security.ssl_mode=2
security.port_non_ssl=8080
security.port_ssl=8443
```

Standard Formats

Batch Date Format

With regard to incoming EDI sent data, retailers can define (through their vendors) the batch date format that they would prefer the system to receive. The system uses the batch date format that the operator enters in this section of the file. For example:

```
batch_date_format=yyyyMMddhhmmss
```

Quantity Decimals Allowed

The database tables within the system allow for quantities to be held in decimals (for example, 12.5). Quantity decimals allowed means how many decimals the system displays for a quantity field. In general, quantity decimals are utilized by grocery retailers.

This is an integer value. ReIM is set to 0 because the system assumes eaches. If a retailer wished to show four decimals, the value would be 4.

Size of Batch Updates

This property establishes the size of the batch updates to the database. An array in this context is a collection of data. The value is in records.

> **Note:** The EDI invoice upload batch process does *not* use this property. For a description of the property that the EDI invoice upload batch process uses, see the section, "Number of New Documents that EDI Invoice Upload Should Insert at a Time," later in this chapter.

For example:

```
# Bulk insert and update array size
ARRAY_PROCESS_SIZE=30
```

Set the End of Week Day for the System

The system administrator establishes this value to inform the system what that end of the weekday is. Sunday is equal to 1, and Saturday is equal to 7.

Batch Log and Error File Paths

Batch Log File Path

The name and directory of the batch log files are established through this setting.

For example:

```
batcherrorlogpath=/files0/ReIM11/dev/error
```

Batch Error File Path

The name and directory of the batch error files are established through this setting. All errors and all routine processing messages for a given program on a given day go into this error file.

For example:

```
batcherrorlogpath=/files0/ReIM11/dev/error
```

Exception Handling and Error Logging Threshold

The ReIMException class automatically logs itself to the application log file. The level of logging may be raised or lowered in the setting below. The operator's choice instructs the system to log that level of error and errors above that level.

In the example below, an operator has configured the system to only display ERROR messages and above:

```
ReIMLoggerLogLevel=ERROR
```

Because the logger reveals message priority levels using numbers, the word equivalents are shown below:

- UNKNOWN = -999
- FATAL = 2
- ERROR = 3
- WARN = 4
- VALIDATION = 5
- INFO = 6
- DEBUG = 7
- PERFORMANCE = 8

Locking Timeout Variable

To commit information to the database, a user must have a "lock" on the database. Locks can time out. The number of seconds until the time out occurs is the "locking timeout variable." If the user no longer has a lock, the user receives a message saying that changes cannot be saved.

The retailer should set the timeout period that makes the most sense for its business needs. Ideally, the timeout period should be long enough so that users can finish working on one record, but short enough so that unintentional locks (during lunch, and so on) do not delay other users an inordinate amount.

During a session, when the system has been idle for longer than the locking timeout variable, the system does not release the lock. Rather, if a second user attempts to lock the same table, the system's locking service determines whether the locking timeout variable has been exceeded. If the locking timeout variable has been exceeded, the locking service continues locking the table but for the second user. For the first user, the lock on the table has expired.

ReIM has a lock table defined for the bulleted areas below. For example, the table IM_DOC_HEAD locks the corresponding values from IM_DOC_HEAD_LOCK. To establish the duration of the timeout, the retailer can enter a mathematical expression using the variables. In the list below, the retailer has set the tables, with one exception, to be released after one hour of session inactivity. The table is only released at the end of the user's session or, in the case of a system crash, at the beginning of the next user's session.

- business_roles_lock_timeout=1*hour
- doc_group_list_lock_timeout=1*hour
- doc_head_lock_timeout=no_expire
- edi_reject_doc_lock_timeout=1*hour
- supplier_options_lock_timeout=1*hour
- system_options_lock_timeout=1*hour
- tolerance_dept_lock_timeout=1*hour
- tolerance_supp_lock_timeout=1*hour

- tolerance_supp_trait_lock_timeout=1*hour
- tolerance_system_lock_timeout=1*hour

Locking timeout variables are in milliseconds. Conversion data is provided below for the retailer's convenience.

- millisecond=1
- second=1000
- hour=3600000
- day=86400000
- month=2592000000
- no_expire=-1

Auto-Match Threading Options

These parameters are used to configure auto-match threading. Auto-match can either be run as a single thread, or it can be threaded by the location hierarchy. Currently, this parameter is defaulted to thread auto-match by district. Changing the thread parameter is as simple as commenting out one parameter and commenting out another.

For example:

```
auto_match_thread_by=ThreadByDistrict
```

Generic Threading Options

For more information about the batch processes mentioned below, see Chapter 7 – "Batch Processes".

Parameter Used by EdiUpload Only

The thread.backgroundThreadTimeout parameter is currently only used by EdiUpload for rejection files. The value represents how long the log writing thread polls an empty work queue before shutting down. Units are expressed in milliseconds.

For example:

```
thread.backgroundThreadTimeout=1800000
```

Parameters Used by EdiUpload, AutoMatch, ComplexDealUpload, and FixedDealUpload

The thread.consumerThreadTimeout parameter represents how long the consumer pool threads. The value is used for executing the transactions for both EdiUpload and AutoMatch. Units are expressed in milliseconds.

For example:

```
thread.consumerThreadTimeout=60000
```

The thread.consumerThreadKeepAlive parameter represents how long the consumer/worker stays alive. Units are expressed in milliseconds.

For example:

```
thread.consumerThreadKeepAlive=60000
```

The thread.consumerThreadPoolMin and thread.consumerThreadPoolMax parameters represent the range of consumer/worker threads that can be created for the pool.

For example:

```
thread.consumerThreadPoolMin=10
thread.consumerThreadPoolMax=100
```

Number of New Documents that EDI Invoice Upload Should Insert at a Time

The EDI invoice upload (ediupinv) uploads merchandise and non-merchandise invoices and credit notes from the EDI into the invoice-matching tables. This parameter, which is related to bulk processing, establishes the number of documents that the system inserts at one time into one or more invoice-matching tables. In the example below, 1000 documents has been established as the value.

For example:

```
#How many new documents should EdiUpload insert at a time
NBR_OF_EDI_DOC_BULK=1000
```

Invoice Characters

Allowable Invoice Characters

This validation-related parameter describes what characters are allowed on an invoice. Note that '\' represents escape characters.

For example:

```
#Invoice number validation regular expression.
#Allowed Characers are :0-9, A-Z, space, minus sign, plus sign and underscore.
#If this property is omitted(commented out) the system will default to =^[0-9A-Z]+$ (only Alpha-Numeric).
INVOICE_NUMBER_VALIDATION_REGULAR_EXPRESSION=^[0-9A-Za-z\ \+\-\_]+$
```

Invoices Beginning with Zero

This parameter either allows or disallows invoices to begin with the number zero. In the example below invoice numbers are *not* allowed to begin with the number zero.

For example:

```
INVOICE_NUMBER_VALIDATION_ALLOW_ZERO=FALSE
```

Deal Detail Purge Parameter

This parameter is related to the ReIM process of extracting deal-related data from RMS. Once a document is posted, this value determines how many days ReIM waits before deleting the detail for the document.

For example:

```
purge_deals_after_days=2
```

system.properties File

This file includes system options settings that were not built into the graphical user interface (GUI) because they cannot be changed once ReIM has been implemented.

Determine Which General Ledger (GL) Options are Dynamic

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. Dynamic segments are those which are driven by location or department and class numbers provided with the invoice (as opposed to being hard-coded). 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 would equal N.

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

- The system allows a maximum of four segments that can be dynamic.
- Those values that are set to dynamic (that is, set to Y) can be associated to the following four concepts (note that company and location are always paired together and that department and class are always paired together):
  - Company
  - Location
  - Department
  - Class
- If the retailer's segments are dynamic by location, the location number is included in the parameter.

  For example:

  ```
  system.gl_option_dynamic_1=Y
  system.gl_option_dynamic_2=Y
  system.gl_option_dynamic_3=N
  system.gl_option_dynamic_4=Y
  system.gl_option_dynamic_5=Y
  system.gl_option_dynamic_6=N
  system.gl_option_dynamic_7=N
  system.gl_option_dynamic_8=N
  system.gl_option_dynamic_9=N
  system.gl_option_dynamic_10=N
  system.gl_option_dynamic_11=N
  system.gl_option_dynamic_12=N
  system.gl_option_dynamic_13=N
  system.gl_option_dynamic_14=N
  #Business concept mapping for dynamic segments
  system.gl_option_dynamic_mapping_1=COMPANY
  system.gl_option_dynamic_mapping_2=LOCATION
  system.gl_option_dynamic_mapping_3=
  system.gl_option_dynamic_mapping_4=DEPARTMENT
  system.gl_option_dynamic_mapping_5=CLASS
  system.gl_option_dynamic_mapping_6=
  system.gl_option_dynamic_mapping_7=
  system.gl_option_dynamic_mapping_8=
  system.gl_option_dynamic_mapping_9=
  system.gl_option_dynamic_mapping_10=
  system.gl_option_dynamic_mapping_11=
  system.gl_option_dynamic_mapping_12=
  system.gl_option_dynamic_mapping_13=
  system.gl_option_dynamic_mapping_14=
  ```

Mapping of Document Types to Action Codes

This is the default action based on document type which limits the reasons presented in the reason code list of values (LOV) on the document maintenance detail screen.

For example:

```
#CREDIT_NOTE_REQUEST_PRICE
CRDNRC=CBC
```

Child Invoice Indicator

In this section, the retailer defines a string. 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 system continues to use the parent invoice ID, along with both the string that is defined by this parameter and the location number to which the child invoice is associated.

For example:

```
system.child_invoice_indicator=LOC
```

Set the Audit Period

The parameter determines how many days the system retains audit trail data.

For example:

```
system.purge_tolerance_audit_period=2
```

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 the section, 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, among others:

- Graphical user interface
- Error messages

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 on the IM_USER_AUTHORIZATION table. The properties files ReIMResources and ReIMMessages 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 '/' (backslash) 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 also the section, "ReIM User Table," in Chapter 5 – "Interfaces and File Layouts."

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, localization/internationalization purposes, and so on).

Translated properties files are shipped in a format that is not human readable. The files have been run through the JDK utility: native2ascii. Therefore, users cannot edit the translated versions of the properties files without first running reverse native2ascii.

IM_USER_AUTHORIZATION

Functionality exists within the system to allow a retailer to change the language displayed in the UI for a specific user. The retailer can write an update statement for the IM_USER_AUTHORIZATION table. The update statement would specify the following for the user name:

▪ A language for a user using the two letter language code (for example, zh, for Chinese)

▪ A country for the user using the two-letter country code (for example, CN, for China)

Once the retailer has run the query, performed a commit, and logged out and into the application, the UI reflects the new language and locale.

> **Note:** The language/locale combination must be valid and supported by the system, or when the retailer logs back into the application, the default language is displayed.

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 provide the "glue" that holds the application together, offering the application frameworks for error logging, internationalization, transaction management, application security, and so on.

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

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

The Layering Model

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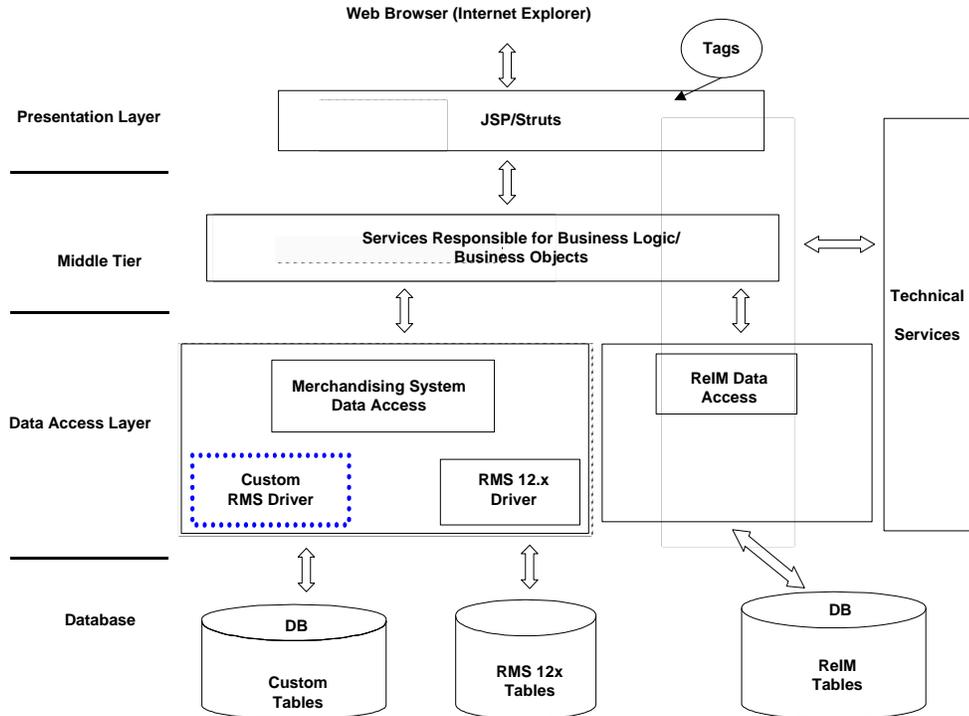
ReIM Layered Architecture

Presentation Layer

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

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

Struts provide an open source framework for building Web applications. The core of Struts is a flexible control layer based upon Java servlets, JavaBeans, 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 only interacts with the middle tier services.

Middle Tier

Service Layer Responsible for Business Logic

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

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

Business Objects

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

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

Data Access Layer (DAL)

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

Database Layer

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

Technical Services

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

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

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

A brief description of each technical service follows the diagram.

Application parameter service	System parameter service	Security service	Log4J
Persistent code generation service	Transaction service		
Error logging service	Internationalization service	Time/date service	
	Currency service		
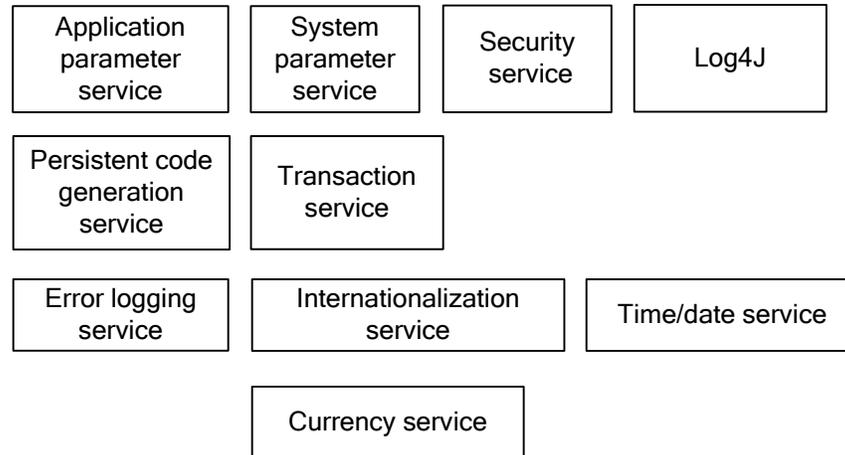
ReIM Technical Services

Application Parameter Service

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

System Parameter Service

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

Transaction Service

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

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

Error Logging Service

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

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

Log4J

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

Internationalization Service

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

Currency Service

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

Time/Date Service

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

Security Service

The security service provides basic authorization and authentication functionality during user 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 the database or LDAP server locations.

Third Party Libraries

ReIM base development uses the following third party libraries:

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

ReIM-Related Java Terms and Standards

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

The Java 2 Enterprise Edition (J2EE)

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

Java Database Connection (JDBC)

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

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

Java Development Kit (JDK)

Standard Java development tools from Sun Microsystems.

Java Server Pages (JSP)

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

Java Servlet

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

LOG4J

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

Naming Conventions in Java

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

Struts

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

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

4

Functional Design

This chapter provides the following:

- An overview as to how ReIM is functionally integrated with other systems (including other Oracle Retail systems). The discussion primarily concerns the flow of ReIM-related business data across the enterprise.
- A diagram and description of the invoice matching process flow

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.
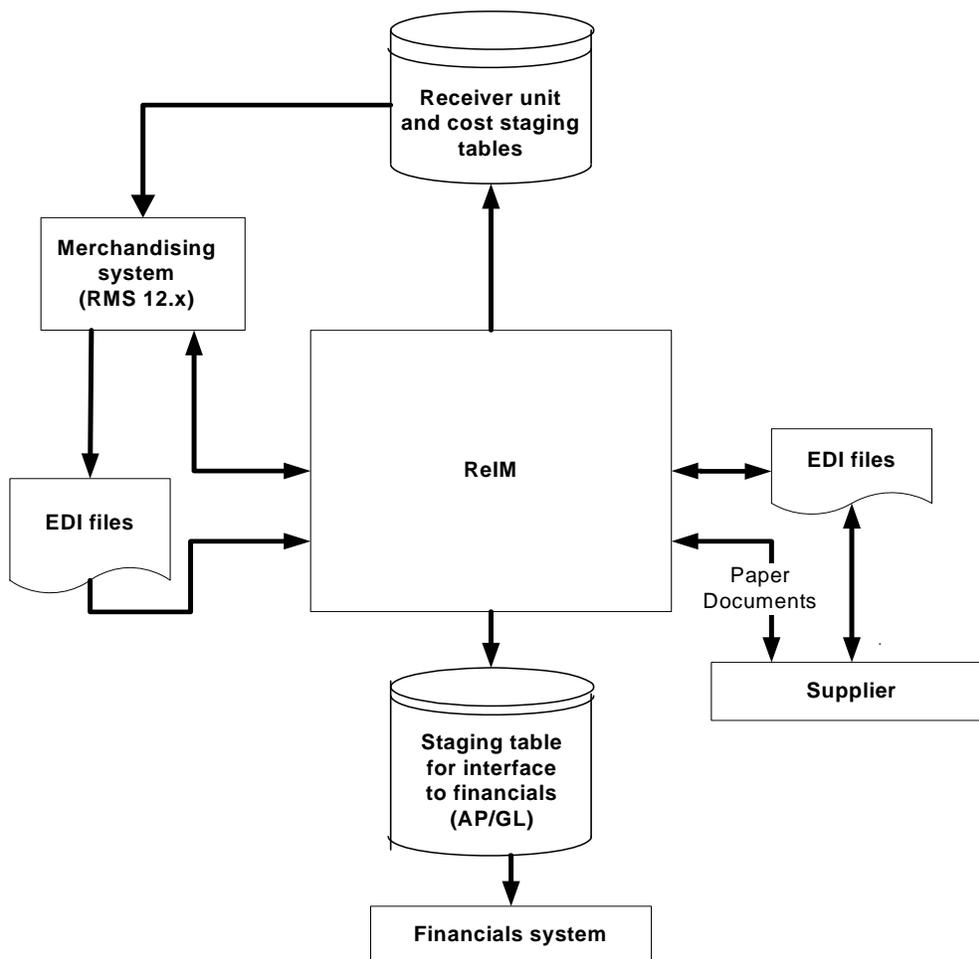
ReIM Data Flow Across the Enterprise

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 (described in Chapter 4). 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 Financial Staging Table

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

From the Merchandising System to ReIM Directly and Through EDI

ReIM can 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 the RMS 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.

- Value added tax (VAT) code and rate
  VAT is embedded in the cost of the item. ReIM provides for validation of VAT taxes charged on the invoice against VAT 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 via 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 (see the discussion later in this chapter)

From ReIM to Receiver Unit and Cost Staging Tables to RMS

Receiver adjustments initiated in ReIM update receipts held in RMS tables. Receiver adjustments, resulting from the ReIM discrepancy resolution process, create cost and/or 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 an 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.

From ReIM to the Financial/AP Staging Tables

ReIM exports data to financial staging tables. If integrated with Oracle Financials, there is a standard interface of data to Accounts Payable and General Ledger. However, if the retailer is using any other financials system, the retailer must create its own interface to deliver this information to the applicable financial system.

Integration with Oracle Financials

When integrated with Oracle e-Business Suite 11.0, ReIM exports data to AP staging tables or to financial staging tables, depending on the specific types of transactions. This is done if the financial system for AP and GL financial systems if Oracle Enterprise Financials and the Oracle Retail Merchandising System (RMS) System Options table have the following settings:

- Financial-AP = O
- Oracle-Financials-Vers = 1

Matched Invoices and Approved Documents

Invoices can be matched through auto-matching or on-line matching. Credit notes can be matched with credit note requests in on-line matching processes. The unit cost and quantities of all items (at a summary level) on the invoice are compared to the unit cost and quantities on the receipt. If the cost and quantity on the invoice and receipt agree within defined tolerances, there is a match.

Pre-Paid Invoices

Invoices may be paid before matching is complete in order to meet payment terms requirements. Users determine whether to pre-pay an invoice. Pre-paid invoices are still eligible for matching against receipts; however, an indicator on the invoice record prevents it from being paid twice. When a pre-paid invoice is matched, the results are posted to the IM_FINANCIALS_STAGE table for interface to the General Ledger rather than the IM_AP_STAGE_HEAD and IM_AP_STAGE_DETAIL tables for interface to Accounts Payable.

Non-Merchandise Invoices

These invoices include bills for non-merchandise costs only. Non-merchandise invoices cannot contain items. Either suppliers or partners can create non-merchandise invoices. However, merchandise invoices can contain non-merchandise lines.

Posting Transaction Codes to AP Staging Table

IM_AP_STAGE_HEAD:

- Invoice Type Lookup Code: If document type = MRCHI or NMRCHI or CRDMEC or CRDMEQ, this value is set to STANDARD. Otherwise this value is set to CREDIT.

IM_AP_STAGE_DETAIL:

The rules for Line Type Lookup Code are as follows:

- If the tran-code is UNR, VWT, VCT, REASON, or CRN, then this value is ITEM.
- If this is a generated tax line, then this value is TAX.
- If none of the above, then this value is MISCELLANEOUS.

Integration with Non-Oracle Financials System

When integrated with a financials system other than Oracle, ReIM exports data to a financial staging table with data intended for both Accounts Payable and General Ledger. The retailer needs to develop their own interface from the financial staging table to their systems, based on the requirements of their financials systems.

Matched Invoices and Approved Documents

Invoices can be matched through auto-matching or on-line matching. Credit notes can be matched with credit note requests in on-line matching processes. The unit cost and quantities of all items (at a summary level) on the invoice are compared to the unit cost and quantities on the receipt. If the cost and quantity on the invoice and receipt agree within defined tolerances, there is a match.

Non-Merchandise Invoices

These invoices include bills for non-merchandise costs only. Non-merchandise invoices cannot contain items. Either suppliers or partners can create non-merchandise invoices. However, merchandise invoices can contain non-merchandise lines.

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
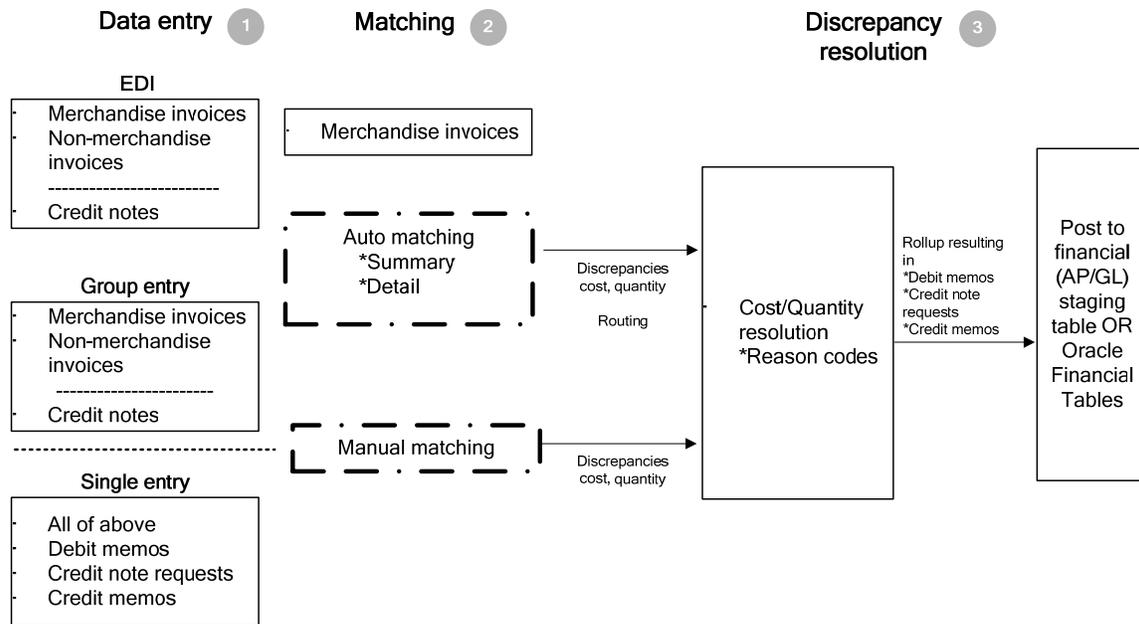
High-Level View of the Invoice and Credit Note Matching Process

> **Note:** Documents drop out of the flow when they need 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.

- **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.

2. **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.

3. **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, because 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 memo
  A document created to refund a supplier for an under-invoiced or over-billed amount (for example, for rebates not meeting threshold performance levels). 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. Doing so ensures related documents are released to accounts payable at the same time.

4. Matching

- Auto-matching

  The auto-matching process attempts to support invoice cost and quantities against receipt quantities at PO cost within user defined tolerances. 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.

  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, VAT 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" section later 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 (unmatched) invoices and receipts, to support matches.

    On-line matching either matches a document posted to the financial staging table, or it 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. 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.

5. 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 its 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.

The Auto-Match Process

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

VAT on Header Level Only Invoices

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

The invoice status determines whether an invoice can be processed by the Auto-match batch process (AutoMatchService). Only invoices in a status of Ready for Match are processed. Those with a status of VAT discrepancy are not processed by the batch. See Chapter 7 – "Batch Processes" for more information.

Invoices created without details are not able to have their VAT information validated at invoice creation. All header level-only invoices are created with a status of Ready for Match. These invoices must have a VAT 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 VAT 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.

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:	**$200,000**	**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**

In the example, even though there is not an exact match, the extended costs match within tolerance (assuming the $135 difference is within summary-level tolerance settings). Therefore, the receipts and invoices are set to matched status.

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
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 3	$35,000	3,000	Multi-unresolved

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	$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
Receipt 3	$35,000	3,000	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	3,000	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
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.

Eligibility 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 PO's unit cost. 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's 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 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
- Receipt line	Item 3	$10.00	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's 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 have 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
Receipt 1		$60,000	1,000	Partially Matched	Matched
- Receipt line	Item 1	$30,000	500	Previously matched	Matched
- Receipt line	Item 2	$15,000	250	Unresolved	Matched
- Receipt line	Item 3	$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	Unit Cost	Quantity	Status Prior to Matching	Status After Matching
Invoice 1				Unresolved	Unresolved
- Invoice line	Item 2	$15.00	250	Unresolved	Unresolved
- Invoice line	Item 3	$15.00	250	Unresolved	Matched

Invoice Lines for a PO/Location	Item	Unit 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.

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 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 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.

Interfaces and File Layouts

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 invoice upload 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 7 – "Batch Processes." Note that although the vast majority of invoices are created through either EDI upload 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.

EDI Reject Table

The EDI invoice upload (ediupinv) batch process uploads invoices and credit notes from the EDI into the invoice-matching tables. This process validates the information in the file against itself and against the RMS (or equivalent merchandising system)/ReIM database. A limited set of data validation errors cause the invalid transaction to be written to error tables (IM_EDI_REJECT_DOC_xxx) where the data can be corrected through an online process.

The following errors are written to the EDI reject table for the user to manually correct 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 v-date minus the post-dated document days' system level parameter value or newer than the v-date.

- 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).

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). When VAT processing is active within ReIM, all failed validations result in EDI uploads being rejected to a file. There are no reject-to-table cases, and the EDI Maintenance screens are not accessible to the retailer.

EDI Invoice Upload File Layout (Based on EDI 810)

I/O Specification

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 optional for Credit Note docs and debit memo docs.

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, optional otherwise.

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 number, 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.

All items must have an associated primary UPC as the EDIUpload File only handles taking in UPCs.

FHEAD – File Header

The file header is the 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

The transaction header is the 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.

Field Name	Field Type	Description	Req	Validation
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 propert,y INVOICE_NUMBER_ VALIDATION_ALLOW_ZERO, in reim.properties is commented out or set to false.

Field Name	Field Type	Description	Req	Validation
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 valid vendor type (from Vendor class) • Document type is MRCHI (merchandise invoice) and vendor type is not Supplier
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 is a supplier and supplier is not valid. • Vendor is a supplier and vendor ID is not completely numeric.
Vendor Document Date	Char(14)	Date document was issued by the vendor (in YYYYMMDDHH24MISS format). Stored in IM_DOC_HEAD.DOC_DATE	Y	Reject transaction to tables if Vendor Document Date is: • After the vdate. • Before (vdate – pos_ dated_ doc_ days) (from im_system_options). Reject transaction to tables if: • Vendor document date is after the vdate or before (vdate – post_dated_doc_days) (from im_system_options).

Field Name	Field Type	Description	Req	Validation
Order Number/RTV order number	Number(10)	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 for 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.

Field Name	Field Type	Description	Req	Validation
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.
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.

Field Name	Field Type	Description	Req	Validation
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 currency of the order.
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.	Y	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 VAT amount.	Y	Reject transaction to file if sign indicator is null or if it is not '+' or '-'.
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 VAT amount is not null but is not numeric. ▪ Total VAT amount does not equal the sum of VAT for all item detail records PLUS the sum of VAT for all non-merch items in this transaction PLUS the sum of VAT for all allowances in this transaction.

Field Name	Field Type	Description	Req	Validation
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total quantity 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 discount 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

Field Name	Field Type	Description	Req	Validation
Consignment indicator	Char(1)	Indicates if this invoice is a consignment invoice.	Y	Reject transaction to file if: ▪ Consignment indicator is null ▪ Consignment indicator is not Y or N Do not reject transaction to table if Consignment is Y.
Deal Id	Number(10)	Deal Id from RMS if this invoice is a deal bill back invoice.	N	If Deal Id is not null, Deal Approval indicator must be M or A. Do not reject transaction to table if deal id is not null.
Deal 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	

Field Name	Field Type	Description	Req	Validation
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_VAT.

Field Name	Field Type	Description	Req	Validation
File record descriptor	Char(5)	Marks costs at VAT 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 VAT 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 VAT 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 entire file if: • Transaction number is not numeric • Transaction number is not the same as the current transaction
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.	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).
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
Original Document 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)	VAT code for item.	Y	Reject to file if VAT code is invalid.
Original VAT rate	Number (20,10)	VAT Rate for the VAT code/item.	Y	Reject to file if VAT rate is not numeric.
Sign Indicator	Char(1)	Indicates either a positive (+) or a negative (-) total allowance. Default is "+" if no allowances exist for this detail record.	Y	Reject transaction to file if: ▪ Sign indicator is null. ▪ Sign indicator is not '+' or '-'

Field Name	Field Type	Description	Req	Validation
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.
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. ALLOWANCE_ 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. ALLOWANCE_ 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 the VAT 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 required only when the document type is non-merchandise. Non-merchandise cost records also are 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 Merchan-dise 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 '-'.
Non Merchan-dise 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.

Field Name	Field Type	Description	Req	Validation
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

The 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

The 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
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:

1. The EDI document upload process only has the ability to recognize 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.

2. For the charge back documents, to decide what document type to be populated in the database, a flow chat is displayed as follows:
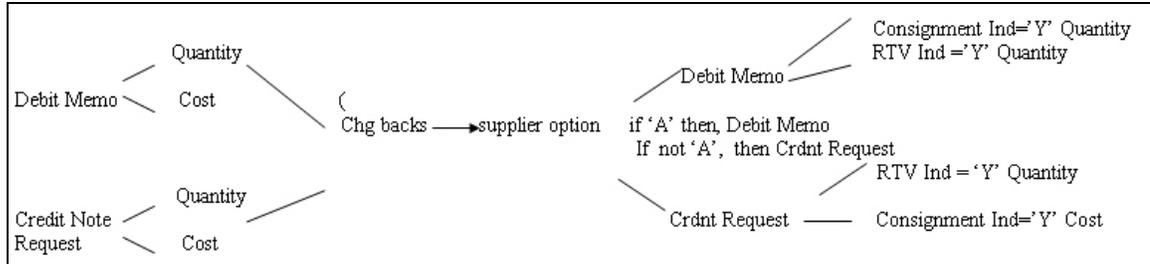
3. If the document type is "merchandise invoice", and if the consignment indicator is Y, the status would be "matched".

   If the consignment indicator is not Y, the status would be "ready for match".

   If the document type is not "merchandise invoice", the status would be "approved".

4. If the consignment indicator is Y, then set the terms to the "Due Immediately" terms (term id = 48), and set the terms discount percentage 0.

5. That VAT 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 VAT. The VAT code and VAT rate in the VAT should be the same as the original VAT code and original VAT rate in the TDETL.

6. The merchandises header VAT and detail VAT are consistent (for example, VAT basis by VAT rate and VAT amount by VAT rate). Total header Merchandise VAT information is calculated from total document VAT information and VAT information for non merchandise costs.

   For example, for each VAT Code in TDETL and TNMRC: Thead.Total VAT Amount at this VAT code = total VAT from TDETL at this VAT code + total VAT from TNMRC at this VAT code.

   Total VAT from TDETL at this VAT code = sum(original document quantity * original unit cost * original VAT rate).

   Total VAT from TNMRC at this VAT code =sum(Non Merch VAT rate * Non Merch Amt).

   Thead.Total VAT Amount at this VAT code = sum(TVATS.VAT rate * TVATS.cost at this VAT code).

7. For an EDI upload document, if the VAT Region of the header is different from the VAT region of the supplier, it is an import document. Import document will not contain VAT 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.

8. To decide whether a VAT code is valid in the TDETL, first find the VAT code given the information of item and location. If they are equal, then the VAT code is valid. If they are not equal, check if the VAT code exists in the effective VAT codes. If the VAT code exists, then it is valid, but it is populated to the audit table.

9. If RTV indicator or consignment indicator is Y and deal id is not null, it must reject to file.

10. 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)

I/O Specification

All Files Layouts Input and Output

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 VAT 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; rather, it is ordered by vendor id (THEAD). It is assumed that this file is broken out by vendor id during the translation process.

FHEAD – File Header

The file header is the 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

The transaction header is the 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	

Field Name	Field Type	Description	Req	Validation
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	Debit memo, credit note request cost, credit note request quantity, credit memos in approved status
Vendor Document Number	Char (30)	Vendor's document number. Retrieved from IM_DOC_HEAD.EXT_DOC_ID.	Y	
Invoice Number	Char(6)	Corresponding invoice resolved by the document. Retrieved from IM_DOC_HEAD.REF_DOC.	Y	
Vendor ID	Number(10)	Vendor for this document. Retrieved from IM_DOC_HEAD.VENDOR.	Y	
Document Date	Char(14)	Date the document was entered into the system in YYYYMMDDHH24MISS format. Retrieved from IM_DOC_HEAD.DOC_DATE.	Y	
Order	Number(10)	Order number for this document, if any. Retrieved from IM_DOC_HEAD.ORDER_NO.	N	
Location	Number(10)	Location for this document, if any. Retrieved from IM_DOC_HEAD.LOCATION.	N	
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.		
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	

Field Name	Field Type	Description	Req	Validation
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	
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.	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.		

Field Name	Field Type	Description	Req	Validation
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)	VAT Rate 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)	VAT Rate corresponding to the VAT code.	Y	

TVATS – VAT Detail Record

Field Name	Field Type	Description	Req	Validation
Record descriptor	Char(5)	TVATS	Y	
Line id	Char(10)	Sequential file 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

The 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	
Transaction 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

The 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 has two types of financial interfaces: foundation financial data and transactional information. Both are described in this section.

Foundation Financial Data

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. ReIM does not provide an interface for this information because it does not directly relate to other information in ReIM. ReIM expects the retailer to directly populate the ReIM location account segments table and keep it in sync with the financial application.

Department/Class Account Segments

ReIM uses department account segments in GL account mappings. ReIM does not provide an interface for this information because it does not directly relate to other information in ReIM. ReIM expects the retailer to directly populate the ReIM department account segment table.

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 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 the section, Financial Posting Batch Design, in Chapter 7 – "Batch Processes."

Major Tables

- IM_DYNAMIC_SEGMENT_DEPT_CLASS
- IM_DYNAMIC_SEGMENT_LOC

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 Name	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 Name	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 Name	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 Name	Type	Nullable
SEQ_NO	NUMBER(10)	N
DOC_ID	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.

Matching and Tracking Receipt Posts Processing

When a match is made, the system creates an IM_RECEIPT_ITEM_POSTING record for each invoice item matched, setting the QTY_MATCHED value to the amount matched. In addition, the system creates an IM_RCPT_ITEM_POSTING_INVOICE record for each invoice matched, setting the status to M. Rather than adding IM_RCPT_ITEM_POSTING_INVOICE records each time a portion of the line is matched, the system creates new sets of records for each match to a receipt item.

With regard to summary match processing, an IM_RCPT_ITEM_POSTING_INVOICE record exists for each invoice for each receipt line item. This record is not used to track which invoice and receipt line are matched, but the record allows the system to detect when to set the QTY_POSTED amount in IM_RECEIPT_ITEM_POSTING. Also, when the system matches at a summary level, all associated records are deleted before current ones are created.

The quantity matched amount is set to either the receipt amount or the resolution amount.

Posting

With regard to the posting process, the system finds each record on the IM_RCPT_ITEM_POSTING_INVOICE table associated with the invoice being posted. When that line is posted, the system changes the status on that table to P. The system then checks whether or not more records exist on that table for the same SEQ_NO. If there are more records, the system engages in no further processing steps. If there not more records, the system sets the QTY_POSTED value to the amount in QTY_MATCHED for that SEQ_NO in IM_RECEIPT_ITEM_POSTING. Because posting can only happen when both the cost and quantity discrepancies are resolved for an invoice, the resolution of cost discrepancies is not tracked.

Once posting is completed, all posted records are moved to the corresponding staging table for each table (IM_RECEIPT_ITEM_POSTING_STAGE and IM_RCPT_ITEM_POSTING_INV_STAGE). The processing involving the staging tables has been designed to enhance performance, so that matching and resolution functionality is not impacted adversely by the receipt tracking functionality.

Reporting

Reporting must be run after the posting batch job has completed. Both ReIM and the merchandising system (such as RMS) must be disabled from user input, and all other batch jobs should be completed or disabled.

To determine the remaining amount available to be posted, all entries for a given receipt item's QTY_POSTED should be rolled up and subtracted from the related SHIPSKU entry. Any receipt write-offs should be added in order to determine the final number remaining against the receipt.

Again, the staging tables, IM_RECEIPT_ITEM_POSTING_STAGE and IM_RCPT_ITEM_POSTING_INV_STAGE, are used in building processes and/or reports against this data. Once posting is completed, these staging tables contain all currently posted entries.

LDAP and Other User Interfaces

There are two types of user authentication supported in ReIM: LDAP and database. A simple switch in the reim.properties file instructs the application as to which method to utilize. See Chapter 2 – "Backend System Administration and Configuration" for more information.

LDAP

LDAP stands for Light Directory Access Protocol. The LDAP standard defines a network protocol for accessing information in a directory.

LDAP is one of the means of user authentication that ReIM supports. If it is used, LDAP is only used within ReIM for user authentication. Because ReIM has specific requirements for ReIM user roles and permissions that are easily retailer configurable, these are defined in the application itself. ReIM reads standard user information from an LDAP server.

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

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.

ReIM User Table

A retailer that does not use LDAP has the option of entering valid users into the ReIM user table. Note, however, that ReIM does not provide any method for inserting user information into the ReIM user table. The retailer is responsible for this interface associated with user information.

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

- When a user that has an active lock exits a screen (that is, the user selects the 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 more 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

- 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

- 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 upload 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

Batch Processes

This chapter provides the following:

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

Batch Architectural Overview

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

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

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

Note the following characteristics of the ReIM batch processes:

- They are not accessible through a graphical user interface.
- They are scheduled by the retailer.
- They are designed to process large volumes of data.

Although ReIM is a system that never stops running, batch processes should be executed when users typically are not using the application (at night, for example).

Generic File

The batches are run using a generic file. The generic file is a template for setting the class path, environment specific configuration. It also defines how batches will be run. Refer to the generic file for complete details.

Class path

```
# reim classpath for java
for FILE in `ls $REIMLIBRARIES`
do
        if [ "$REIMCLASSPATH" = "" ];then
                REIMCLASSPATH=$REIMLIBRARIES/$FILE
        else
                REIMCLASSPATH=$REIMCLASSPATH:$REIMLIBRARIES/$FILE
        fi
done

REIMCLASSPATH=$REIMCLASSPATH:$REIMCLASSES
REIMCLASSPATH=$ORACLE_HOME/jdbc/lib/ojdbc14.jar:$REIMCLASSPATH
```

OS specific settings

```
# need to set library paths for use of JDBC thick client
# Solaris and Linux: LD_LIBRARY_PATH
# AIX: LIBPATH
# HP-UX: SHLIB_PATH
if [ -d "$ORACLE_HOME/lib32" ] ; then
    LD_LIBRARY_PATH=$ORACLE_HOME/lib32:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH
    LIBPATH=$ORACLE_HOME/lib32:$LIBPATH; export LIBPATH
    SHLIB_PATH=$ORACLE_HOME/lib32:$SHLIB_PATH; export SHLIB_PATH
else
    LD_LIBRARY_PATH=$ORACLE_HOME/lib:$LD_LIBRARY_PATH; export LD_LIBRARY_PATH
    LIBPATH=$ORACLE_HOME/lib:$LIBPATH; export LIBPATH
    SHLIB_PATH=$ORACLE_HOME/lib:$SHLIB_PATH; export SHLIB_PATH
Fi
```

Execute Batch Jobs Using Arguments from the Merchandising Batch Schedule

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

EDI-Related File-Based Batch Processes

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

Internal Batch Processes

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

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

- Auto-match
- Batch purge
- Discrepancy purge
- Disputed credit memo action rollup
- Reason code action rollup

Internal Batch Processes that Write to Staging Tables

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

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

The internal batch processes that write to staging tables are described later in this chapter. They include the following:

- FinancialPosting
- 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 and Java Packages

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

Batch Name	Class (oracle.retail.reim.batch.jobs)	Program Name (Merchandising Batch Schedule)
Batch purge	BatchPurgeBatch	reimpurge
Discrepancy purge	DiscrepancyPurgeBatch	reimdiscrepancypurge
EDI invoice upload	EdiUploadBatch	reimediinvupload
Auto-match	AutoMatchBatch	reimautomatch
Receipt write-off	ReceiptWriteOffBatch	reimreceiptwriteoff
Reason code action rollup	ReasonCodeActionRollupBatch	reimrollup
Disputed credit memo action rollup	DisputedCreditMemoResolutionRollupBatch	reimrollup
Financial posting	FinancialPostingBatch	reimposting
EDI invoice download	EdiDownloadBatch	reimediinvdownload
Complex deal upload	ComplexDealUploadBatch	reimcomplexdealupload
Fixed deal upload	FixedDealUploadBatch	reimfixeddealupload

Note: "reimrollup" is the common program to execute batches, "Reason code action rollup" and "Disputed credit memo action rollup." The reimrollup program executes the batches in the following sequence: 1. Reason code action rollup batch; 2. Disputed credit memo action rollup batch.

Functional Descriptions and Dependencies

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

Batch Processes	Details	Batch Dependencies
Batch purge	This process deletes data from database tables while maintaining database integrity. This process deletes records from the ReIM application that meet certain business criteria (for example, records that are marked for deletion by the application user, records that linger in the system beyond certain number of days, and so on).	
Discrepancy purge	The discrepancy purging program deletes data from database tables while maintaining database integrity. This program deletes records from ReIM that have discrepancies of zero.	
EDI invoice upload	This batch process uploads merchandise, non-merchandise invoices, credit notes, debit memos, and credit note requests from the EDI into the invoice-matching tables.	
Auto-match	Auto-match is a system batch process that attempts to match invoices to receipts without manual intervention. Invoices that are in ready for match, unresolved, or multi-unresolved status are retrieved from the database to be run through the auto-match algorithm. The processing consists of three levels – summary, detail, and header (VAT only).	▪ EDI upload (Invoice Matching) ▪ Receipt upload (Merchandising system, such as RMS)
Receipt write-off	In order for retailers to track received goods not invoiced, they must have the ability to "write-off" these goods for financial tracking. ReIM has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the Receipt write-off process is run, each non-fully matched open receipt received date is compared with the current date minus the system parameter. If the received date is before this difference, the receipt is written-off, and the invoice match status is closed.	Auto-match and any associated processing must run prior to this batch processing.

Batch Processes	Details	Batch Dependencies
Reason code action rollup	This batch process sweeps the action staging table and creates debit and credit memos as needed. Only a single debit or credit memo is created per invoice/discrepancy type, with line details from all related actions for the same discrepancy type. This process deletes these records when completed; they are deleted after posting. Note that a separate, retailer-created batch process sweeps the receiver adjustment table. The action staging table is used during posting to post the reason code actions to the financial staging table. A separate, retailer-created batch process sweeps the receiver adjustment table. The process compares the unit cost and/or quantity received for the item on the shipment with the expected unit cost and/or quantity on the IM_RECEIVER_COST_ADJUST and/or IM_RECEIVER_UNIT_ADJUST tables. If a match exists, the receiver cost and/or unit adjustment has occurred in RMS (or the equivalent merchandising system). As a result, the process sets the "pending adjustment" flag on IM_INVOICE_DETAIL table to false for the invoice line. The reason code actions are only rolled up for an invoice if no invoice lines on the invoice have any pending adjustments.	
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 financial posting and after receiver adjustment.
Financial posting	A recurring financial posting process retrieves all matched invoices and approved documents. For each invoice, the batch process writes applicable financial accounting transactions to 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 shows FINANCIAL-AP = O and Oracle-Financials-Vers = 1	
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 financial posting process and creates a flat file. The client converts the flat file into an EDI format by the client and sends it via the EDI invoice download transaction set.	Auto-match must run prior to the EDI invoice download.

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

Features of the Batch Processes

Scheduler and the Command Line

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

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

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

Batch Return Values

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

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

Batch Log and Error File Paths

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

Multi Threading Batch Processes

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

Complex Deal Upload (ComplexDealUploadBatch)

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

Fixed Deal Upload (FixedDealUploadBatch)

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

EDI Invoice Upload (ediupinv)

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

Auto-Match (AutoMatchService)

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

Restart and Recovery

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

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

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

Executing Batch Processes

For the process used to run ReIM batches, a common class has been added. This new common class is represented by the classpath within the generic file used to execute ReIM batch processes. The script in the generic file is changed so that it will add the required jar files into classpath from WEB-INF/lib folder of ReIM home. If a script is used to run batches, retailers also must make the same change to the classpath within that script.

Once the updates are made to the generic file and to the script, batches can be run using the appropriate arguments for each job, as defined in the Merchandising Batch Schedule.

For example, to run the ReIM batch process called AutoMatchBatch, the command is as follows:

```
scriptnameusedtorunbatches <<Batch name as per the merchandise batch schedule
document>> username/password
```

Batch Purge Batch Design

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

Usage

The following arguments are applicable for the BatchPurgeBatch process:

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

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

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

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

SQL Queries

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

Manual Propagation (Cascade) of Deletes to Child Tables

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

Cascade Relationships

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

Step 1

```
Delete from: IM_DETAIL_MATCH_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_DOC_VAT
```

Step 3

```
Delete from: IM_DOC_HEAD
```

Cascade relationships are wired in the BatchPurge.java.

Assumptions and Scheduling Notes

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

Major Modules

BatchPurge

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

Primary Tables Involved

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

- IM_DOC_GROUP_LIST
- IM_DOC_HEAD
- IM_PARENT_INVOICE
- IM_REASON_CODES

Other tables of less significance also get purged.

Discrepancy Purge Batch Design

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

- Usage

  The following arguments are applicable for the DiscrepancyPurgeBatch process:

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

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

  – IM_COST_DISCREPANCY
  – IM_QTY_DISCREPANCY

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

- SQL Queries

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

Major Modules

DiscrepancyPurge

Major Tables

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

EDI Invoice Upload Batch Design

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

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

Assumptions and Scheduling Notes

- This process must be run before the auto-match process.
- All quantities are assumed to be in eaches (EA) when uploaded.

Restart and Recovery

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

Primary Tables Involved

- IM_DOC_HEAD
- IM_INVOICE_DETAIL
- IM_INVOICE_DETAIL_ALLOWANCE
- IM_DOC_NON_MERCH
- IM_DOC_DETAIL_REASON_CODES
- IM_PARENT_INVOICE
- IM_PARENT_INVOICE_DETAIL
- IM_PARENT_NON_MERCH
- IM_EDI_REJECT_DOC_DETAIL
- IM_EDI_REJECT_DOC_DETAIL_ALLOW
- IM_EDI_REJECT_DOC_HEAD
- IM_EDI_REJECT_DOC_NON_MERCH
- IM_DOC_VAT

Auto-Match Batch Design

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

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

- 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 VAT before continuing to attempt to match all invoices.

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

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

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

Algorithms

The auto-match process is composed of the following algorithms:

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

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

3. **One-to-one invoice matching**

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

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

4. **Detail matching**

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

5. **Header matching (VAT only)**

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

Assumptions and Scheduling Notes

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

  The unit costs for the invoice and for the PO must be expressed in the same currency. To compare invoice unit costs with PO unit costs, auto-match does not engage in currency conversion.

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

- The quantities on the invoice must be expressed in the same unit of measure as the quantities on the receipt. Auto-match performs no unit of measure conversion.

- The batch process runs after EDI upload (Invoice Matching) and Receipt upload (Merchandising system, such as RMS).

- Supplier options

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

Post Processing

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

- Auto-match automatically posts invoices that it matches.

High-Level Flow Diagram

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

ReIM Auto-Match Flow

Primary Tables Involved

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

Receipt Write-Off Batch Design

Retailers track received goods that are not invoiced, and they must have the ability to "write-off" these goods for financial tracking. Two types of processes can determine when these written-off goods will be written to financials: purged receipts from merchandising system, and "close open receipts" from invoice matching. Because receipts can be purged outside of the invoice matching dialogue, these purged receipts must be maintained until their unmatched amount has been accounted for. These receipts are tracked through STAGE_PURGED_SHIPMENTS and STAGE_PURGED_SHIPSKUS. Every purged shipment record that is not fully matched will have a record by item written to the stage tables. In addition, invoice matching has a system parameter (which can be overwritten at the supplier level) defining the maximum amount of time an open, non-fully matched receipt will be available for matching. Every time the write-off process is 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:

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

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

> **Note:** If IM_SUPPLIER_OPTIONS.CLOSE_OPEN_
> RECEIPT_MONTHS is not defined, the value is retrieved
> from IM_SYSTEM_OPTIONS.CLOSE_OPEN_
> RECEIPT_MONTHS.

Assumptions and Scheduling Notes

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

- Auto-match and any associated processing must be run prior to this batch processing.

High-Level Flow Diagram

Primary Tables Involved

ReIM

- IM_FINANCIALS_STAGE
- IM_SYSTEM_OPTIONS
- IM_SUPPLIER_OPTIONS
- IM_PARTIALLY_MATCHED_RECEIPTS

RMS

- UNIT_OPTIONS
- SHIPMENT
- STAGE_PURGED_SHIPMENT
- SHIPSKU
- STAGE_PURGED_SHIPSKU

Reason Code Action Rollup Batch Design

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

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

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

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

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

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

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

The process compares the unit cost and/or quantity received for the item on the shipment with the expected unit cost and/or quantity on the IM_RECEIVER_COST_ADJUST and/or IM_RECEIVER_UNIT_ADJUST tables. If a match 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.

Assumptions and Scheduling Notes

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

High-Level Flow Diagram

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

ReIM Reason Code Action Rollup Flow

Primary Tables Involved

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

Disputed Credit Memo Action Rollup Batch Design

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

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

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

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

Assumptions and Scheduling Notes

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

Primary Tables Involved

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

- IM_DOC_HEAD

  This table holds the document header information.

- IM_DOC_DETAIL_REASON_CODES

  This table holds the document detail information by item/reason code. Before resolution rollup, this table holds the document detail information based on the original debit reason codes. After resolution rollup, this table holds the document detail information based on the reason codes used to resolve the disputed credit memo lines.

- IM_REVERSAL_RESOLUTION_ACTION

  This table holds the resolution actions the user takes to approve or deny the disputed credit memo line.

- IM_COST_DISCREPANCY

  This table holds the disputed credit memo lines for a debit memo cost reversal.

- IM_QTY_DISCREPANCY

  This table holds the disputed credit memo lines for a debit memo quantity reversal.

- IM_QTY_DISCREPANCY_ROLE

  This table holds the routing information for a credit memo quantity.

Financial Posting Batch Design

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

- The Financials staging table, IM_FINANCIALS_STAGE

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

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

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

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

Assumptions and Scheduling Notes

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

- Set up segment definitions in the system.properties.

- Define GL account segments on the GL Options screen.

- Specify all the accounts using the GL Cross Reference screen.

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

- Country

- Location

- Dept

- Class

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

Primary Tables Involved

- IM_DOC_HEAD

  Holds the matched and approved documents.

- IM_DOC_NON_MERCH

  Holds the non-merchandise costs for invoices.

Lookup Tables that must be Populated

- IM_GL_OPTIONS

  Order of segments and dynamic segments defined.

- IM_GL_CROSS_REF

  Account values defined for account types and account codes.

- IM_DYNAMIC_SEGMENT_DEPT_CLASS

  Accounts defined for each department/class combination.

- IM_DYNAMIC_SEGMENT_LOC

  Accounts defined for each location/company combination.

Table to which the Process Posts Data

IM_FINANCIALS_STAGE

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

OR

IM_AP_STAGE_HEAD

- Sequence Number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID; for identification purpose.
- Doc_id: Similar to IM_FINANCIALS_STAGE.
- Invoice Type Lookup Code: If document type = MRCHI or NMRCHI, this value is set to STANDARD. Otherwise this value is set to CREDIT.
- invoice_number: The concatenated data is as follows:
  - chars 1-34: the first 34 characters from the EXT DOC ID
  - char 35: a hyphen
  - chars 36-50: the DOC ID

- Vendor: Same as for current im staging table.
- Oracle_site_id:
  - The loc from this transaction to read new RMS Location/Org Unit data to find the Org Unit.
  - The Org Unit to read new RMS Supplier Addr/Org Unit/Site ID data to find Oracle Site ID.
- Currency Code: Valued if this is a foreign currency invoice, otherwise null.
- Exchange Rate: If exchange rate is valued, this should be the literal USER; otherwise blank.
- Exchange Rate Type:
- Document Date: Same as in current im staging table.
- Amount: The TOTAL amount including tax.
- Best Terms Date: Same as in current im staging table.
- Segment1: Same as in current IM financials staging table.
- Segment2: Same as in current IM financials staging table.
- Segment3: Same as in current IM financials staging table.
- Segment4: Same as in current IM financials staging table.
- Segment5: Same as in current IM financials staging table.
- Segment6: Same as in current IM financials staging table.
- Segment7: Same as in current IM financials staging table.
- Segment8: Same as in current IM financials staging table.
- Segment9: Same as in current IM financials staging table.
- Segment10: Same as in current IM financials staging table.
- Create Date: Same as in current IM financials staging table.
- Best Terms ID: Same as in current IM financials staging table.

IM_AP_STAGE_DETAIL

- Doc_id
- Sequence number: Automatically generated line numbers 1, 2, 3, and so on; incremented for each detail record per DOC ID; for identification purpose.
- Transaction Code
- Line Type Lookup Code: This value varies. The rules are:
  - If the tran-code is UNR or VWT or VCT or REASON or CRN then this value is ITEM.
  - If this is a generated tax line, then this value will be TAX.
  - If none of the above, then this value will be MISCELLANEOUS.
- Amount
- VAT Code: Same as in current im staging table. EXCEPT - for generated tax lines, the amount for this line should be the amount from the taxable line times the tax rate
- Segment1: For regular lines: same as in current staging table. For generated tax line: use values from source line.
- Segment2: (see rules for segment 1)
- Segment3: (see rules for segment 1)
- Segment4: (see rules for segment 1)
- Segment5: (see rules for segment 1)

- Segment6: (see rules for segment 1)
- Segment7: (see rules for segment 1)
- Segment8: (see rules for segment 1)
- Segment9: (see rules for segment 1)
- Segment10: (see rules for segment 1)
- Create Date: Same as in current IM staging table.

EDI Invoice Download Batch Design

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

Assumptions and Scheduling Notes

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

Primary Tables Involved

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

- IM_DOC_HEAD
- IM_DOC_DETAIL_REASON_CODES
- IM_DOC_NON_MERCH
- IM_DOC_DETAIL_COMMENTS

Restart and Recovery

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

Complex Deal Upload Batch Design

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

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

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

Assumptions and Scheduling Notes

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

Primary Tables Involved

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

- STAGE_COMPLEX_DEAL_HEAD (RMS table)
- STAGE_COMPLEX_DEAL_DETAIL (RMS table)
- IM_DOC_HEAD

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

- IM_DOC_DETAIL_REASON_CODES

  This table contains quantity/unit cost adjustments for a given document/item/reason code.

- IM_DOC_VAT

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

- IM_COMPLEX_DEAL_DETAIL

  This table holds the details of the complex deal stored in ReIM. It is used during complex deal detail posting.

Fixed Deal Upload Batch Design

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

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

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

Assumptions and Scheduling Notes

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

Primary Tables Involved

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

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

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

- IM_DOC_NON_MERCH

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

- IM_DOC_VAT

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

- IM_FIXED_DEAL_DETAIL

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

8

RETL Program Overview for the ReIM
Extraction Program

To facilitate the extraction of data from ReIM (that could be eventually loaded into a data warehouse for reporting purposes, for example), ReIM works in conjunction with the Retail Extract Transform and Load (RETL) framework. This architecture optimizes a high performance data processing tool that can let database batch processes take advantage of parallel processing capabilities.

Oracle Retail streamlined RETL code provides for less data storage, easier implementation, and reduced maintenance requirements through decreased code volume and complexity. The RETL scripts are Korn shell scripts that are executable from a Unix prompt. A typical run and debugging situation is provided later in this chapter.

These extractions were initially designed for Retail Data Warehouse (RDW) but can be used for some other application in the retailer's enterprise.

For more information about the RETL tool, see the latest RETL Programmer's Guide.

Architectural Design

The diagram below illustrates the extraction processing architecture for ReIM. Instead of managing the change captures as they occur in the source system during the day, the process involves extracting the current data from the source system. The extracted data is output to flat files. These flat files are then available for consumption by a product such as Retail Data Warehouse (RDW).

The target system, (RDW, for example), has its own way of completing the transformations and loading the necessary data into its system, where it can be used for further processing in the environment.

ReIM modules use the same libraries, resource files, and configuration files as RMS. All these libraries, resource files, and configure files are packed with RMS. ReIM must have RMS installed before any ReIM RETL scripts can be kicked off.

ReIM Extraction Architecture

The architecture relies upon the use of well-defined flows specific to the ReIM database. The resulting output is comprised of data files written in a well-defined schema file format. This extraction includes no destination specific code.

RETL Extraction Processing for ReIM

Configuration

RETL

Before trying to configure and run ReIM ETL, install RETL version 12.0 or later, which is required to run ReIM 12.0 RETL. Run the verify_retl script (included as part of the RETL installation) to ensure that RETL is working properly before proceeding.

RETL User and Permissions

ReIM ETL is installed and run as the RETL user. Additionally, the permissions are set up as per the RETL Programmer's Guide. ReIM ETL reads data, creates, deletes, and updates tables. If these permissions are not set up properly, extractions fail.

Environment Variables

See the RETL Programmer's Guide for RETL environment variables that must be set up for your version of RETL. You will need to set MMHOME to your base directory for ReIM RETL. This is the top level directory that you selected during the installation process. In your .kshrc, you should add a line such as the following:

```
export MMHOME=<base directory for RMS ETL>\dwi12.0\dev
```

> **Note:** Because ReIM modules share the same libraries and configuration files as RMS, MMHOME is the same as what is defined in RMS.

dwi_config.env Settings

Make sure to review the environmental parameters in the dwi_config.env file before executing batch modules. There are several variables you must change depending upon your local settings:

For example:

```
export DBNAME=int9i
export RIM_OWNER=steffej_reim1102
export BA_OWNER=rmsint1102
export ORACLE_PORT="1524"
export ORACLE_HOST="mspdev38"
```

You must set up the environment variable PASSWORD in dwi_config.env. In the example below, adding the line to the dwi_config.env causes the password, mypasswd, to be used to log into the database:

```
export PASSWORD=mypasswd
```

Steps to Configure RETL

1. Log in to the UNIX server with a UNIX account that will run the RETL scripts.

2. Change directories to `$MMHOME/rfx/etc`.

3. Modify the dwi_config.env script:

   a. Change the DBNAME variable to the name of the ReIM database.

   b. Change the RIM_OWNER variable to the username of the ReIM schema owner.

   c. Change the BA_OWNER variable to the username of the ReIME batch user.

   d. Change the ORACLE_HOST variable to the database server name.

   e. Change the ORACLE_PORT variable to the database port number

   f. Change the MAX_NUM_COLS variable to modify the maximum number of columns from which RETL selects records.

   > **Note:** All ReIM tables must be under the RMS database. ReIM has the same BA_OWNER as RMS. Thus, the only piece that ReIM modifies in dwi_config.env file is to assign a value to RIM_OWNER. The configuration file, dwi_config.env, as well as all other configuration files, are packed with RMS.

Program Features

RETL programs use one return code to indicate successful completion. If the program successfully runs, a zero (0) is returned. If the program fails, a non-zero is returned.

Program Status Control Files

To prevent a program from running while the same program is already running against the same set of data, the ReIME code utilizes a program status control file. At the beginning of each module, dwi_config.env is run. It checks for the existence of the program status control file. If the file exists, then a message stating, "${PROGRAM_NAME} has already started", is logged and the module exits. If the file does not exist, a program status control file is created and the module executes.

If the module fails at any point, the program status control file is not removed, and the user is responsible for removing the control file before re-running the module.

File Naming Conventions

The naming convention of the program status control file allows a program whose input is a text file to be run multiple times at the same time against different files.

The name and directory of the program status control file is set in the configuration file (dwi_config.env). The directory defaults to $MMHOME/error. The naming convention for the program status control file itself defaults to the following dot-separated file name:

- The program name
- The first filename, if one is specified on the command line
- status
- The business virtual date for which the module was run

For example, the program status control file for the invildex program would be named as follows for the VDATE of March 21, 2004:

```
$MMHOME/error/sincildex.sincilddm.txt.status.20040321
```

Restart and Recovery

Because RETL processes all records as a set, as opposed to one record at a time, the method for restart and recovery must be different from the method that is used for Pro*C. The restart and recovery process serves the following two purposes:

- It prevents the loss of data due to program or database failure.
- It increases performance when restarting after a program or database failure by limiting the amount of reprocessing that needs to occur.

The ReIM extract module (ReIME) extracts from a source transaction database writes to a text file.

To limit the amount of data that needs to be re-processed, more complicated modules that require the use of multiple RETL flows utilize a bookmark method for restart and recovery. This method allows the module to be restarted at the point of last success and run to completion. The bookmark restart/recovery method incorporates the use of a bookmark flag to indicate which step of the process should be run next. For each step in the process, the bookmark flag is written to and read from a bookmark file.

> **Note:** If the fix for the problem causing the failure requires changing data in the source table or file, then the bookmark file must be removed and the process must be re-run from the beginning in order to extract the changed data.

Bookmark File

The name and directory of the restart and recovery bookmark file is set in the configuration file (dwi_config.env). The directory defaults to $MMHOME/rfx/bookmark. The naming convention for the bookmark file itself defaults to the following dot-separated file name:

- The program name
- The first filename, if one is specified on the command line
- bkm
- The business virtual date for which the module was run

The example below illustrates the bookmark flag for the invildex program run on the VDATE of January 5, 2004:

`$MMHOME/rfx/bookmark/sincildex.sincilddm.txt.bkm.20040105`

Message Logging

Message logs are written daily in a format described in this section.

Daily Log File

Every RETL program writes a message to the daily log file when it starts and when it finishes. The name and directory of the daily log file is set in the configuration file (dwi_config.env). The directory defaults to $MMHOME/log. All log files are encoded UTF-8.

The naming convention of the daily log file defaults to the following dot-separated file name:

- The business virtual date for which the modules are run
- .log

For example, the location and the name of the log file for the business virtual date (VDATE) of March 21, 2004 would be the following:

`$MMHOME/log/20040321.log`

Format

As the following examples illustrate, every message written to a log file has the name of the program, a timestamp, and either an informational or error message:

```
sincildex 12:51:07: Program starting...
sincildex 12:51:07: last max post date is 20010311000000
sincildex 12:51:07: Retrieve current max post date
sincildex 12:51:10: Loading invc_exchng_rate_temp table ...
sincildex 12:51:15: Loading po_exchng_rate_temp table ...
sincildex 12:51:20: Process all records between last post date and current max
post date
sincildex 12:51:27: Drop table rmsint110buser1.INVC_EXCHNG_RATE_TEMP
sincildex 12:51:27: Drop table rmsint110buser1.PO_EXCHNG_RATE_TEMP
sincildex 12:51:27: Number of records in sincilddm.txt = 15
sincildex 12:51:27: Program completed successfully
```

If a program finishes unsuccessfully, an error file is usually written that indicates where the problem occurred in the process. There are some error messages written to the log file, such as "No output file specified", that require no further explanation written to the error file.

Program Error File

In addition to the daily log file, each program also writes its own detail flow and error messages. Rather than clutter the daily log file with these messages, each program writes out its errors to a separate error file unique to each execution.

The name and directory of the program error file is set in the configuration file (dwi_config.env). The directory defaults to $MMHOME/error. All errors and *all routine processing messages* for a given program on a given day go into this error file (for example, it will contain both the stderr and stdout from the call to RETL). All error files are encoded UTF-8.

The naming convention for the program's error file defaults to the following dot-separated file name:

- The program name
- The first filename, if one is specified on the command line
- The business virtual date for which the module was run

For example, all errors and detail log information for the invildex program would be placed in the following file for the batch run of March 21, 2004:

```
$MMHOME/error/sincildex.sincilddm.txt.20040321
```

ReIME Reject Files

The ReIME extract module may produce a reject file if it encounters data related problems, such as an inability to find data on required lookup tables. The module tries to process all data and then indicates that records were rejected so that all data problems can be identified in one pass and corrected; then, the module can be re-run to successful completion. If a module does reject records, the reject file is *not* removed, and the user is responsible for removing the reject file before re-running the module.

The records in the reject file contain an error message and key information from the rejected record. The following example illustrates a record that is rejected due to problems within the currency conversion library:

```
Unable to convert currency for LOC_IDNT, DAY_DT|3|20011002
```

The name and directory of the reject file is set in the configuration file (dwi_config.env). The directory defaults to $MMHOME/data.

> **Note:** A directory specific to reject files can be created. The dwi_config.env file would need to be changed to point to that directory.

Schema Files

RETL uses schema files to specify the format of incoming or outgoing datasets. The schema file defines each column's data type and format, which is then used within RETL to format/handle the data. For more information about schema files, see the latest RETL Programmer's Guide. Schema file names are hard-coded within each module since they do not change on a day-to-day basis. All schema files end with ".schema" and are placed in the "rfx/schema" directory.

Resource Files

The ReIM Kornshell program uses resource files so that the same RETL program can run in various language environments. For each language, there is one resource file.

Resource files contain hard-coded strings that are used by extract programs. The name and directory of the resource file is set in the configuration file (dwi_config.env). The default directory is ${MMHOME}/rfx/include.

The naming convention for the resource file follows the two-letter ISO code standard abbreviation for languages (for example, en for English, fr for French, ja for Japanese, es for Spanish, de for German, and so on).

> **Note:** Resource files are packed only with RMS.

Command Line Parameters

The module handles command line parameters in the way described in this section. See the section, RETL Extraction Program List, to determine the command line parameters for a module.

> **Note:** For some RETL modules across Oracle Retail products, default output file names, and schema names correspond to RDW program names.

A Non-File Based Module that Requires Parameters

In order for the non-file based RETL module to run, command line parameters need to be passed in at the UNIX command line. This ReIME module requires an output_file_path and output_file_name to be passed in. This module may allow the operator to specify more than one output file.

For example:

```
sincildex.ksh output_file_path/output_file_name
```

Typical Run and Debugging Situations

The following examples illustrate typical run and debugging situations for types of programs. The log, error, and so on file names referenced below assume that the module is run on the business virtual date of March 9, 2004. See the previously described naming conventions for the location of each file.

For example:

To run sincildex.ksh:

1. Change directories to $MMHOME/rfx/src.

2. At a UNIX prompt enter:

```
%sincildex.ksh $MMHOME/data/sincilddm.txt
```

If the module runs successfully, the following results:

- **Log file:** Today's log file, 20040309.log, contains the messages "Program started …" and "Program completed successfully" for sincildex.ksh.
- **Data:** The sincilddm.txt file exists in the $MMHOME/data directory and contains the extracted records.
- **Error file:** The program's error file, sincildex.sincilddm.txt.20040309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no additional error messages.
- **Program status control:** The program status control file, sincildex.sincilddm.txt .status.20040309, does not exist.
- **Reject file:** The reject file, sincildex.sincilddm.txt.rej.20040309, does not exist.

If the module does *not* run successfully, the following results:

- **Log file:** Today's log file, 20040309.log, does not contain the "Program completed successfully" message for sincildex.ksh.
- **Data:** The sincilddm.txt file may exist in the data directory but may not contain all the extracted records.
- **Error file:** The program's error file, sincildex.sincilddm.txt.20040309, may contain an error message.
- **Program status control:** The program status control file, sincildex.sincilddm.txt.status.20040309, exists.
- **Reject file:** The reject file, sincildex.sincilddm.txt.rej.20040309, does not exist because this module does not reject records.
- **Bookmark file (in certain conditions):** The bookmark file, sincildex.sincilddm.txt.bkm.20040309, exists because this module contains more than one flow. The error occurred after the first flow (for example, during the second flow).

To rerun a module from the beginning, perform the following actions:

1. Determine and fix the problem causing the error.
2. Remove the program's status control file.
3. Remove the bookmark file from $MMHOME/rfx/bookmark
4. Change directories to $MMHOME/rfx/src. At a UNIX prompt, enter:

   ```
   %sincildex.ksh $MMHOME/data/sincilddm.txt
   ```

   > **Note:** To understand how to engage in the restart and recovery process, see the section, "Restart and Recovery," earlier in this chapter.

RETL Extraction Program List

This section serves as a reference to the RETL extraction ReIM program.

Program	Functional Area	Source Table or File	Schema File	Target File	Arguments
sincildex.ksh	Supplier Invoice Cost	IM_DOC_HEAD, IM_INVOICE_DETAIL, ORDLOC, ITEM_MASTER	sincilddm. schema	sincilddm.txt	output_file_path/file name

Application Programming Interface (API) Flat File Specifications

This section contains APIs that describe the file format specifications for all text files.

In addition to providing individual field description and formatting information, the APIs provide basic business rules for the incoming data.

API Format

Each API contains a business rules section and a file layout. Some general business rules and standards are common to all APIs. The business rules are used to ensure the integrity of the information held within RDW. In addition, each API contains a list of rules that are specific to that particular API.

File Layout

- Field Name: Provides the name of the field in the text file.
- Description: Provides a brief explanation of the information held in the field.
- Data Type/Bytes: Includes both data type and maximum column length. Data type identifies one of three valid data types: character, number, or date. Bytes identifies the maximum bytes available for a field. A field may not exceed the maximum number of bytes (note that ASCII characters usually have a ratio of 1 byte = 1 character)
  - Character: Can hold letters (a,b,c…), numbers (1,2,3…), and special characters ($,#,&…)
  - Numbers: Can hold only numbers (1,2,3…)
  - Date: Holds a specific year, month, day combination. The format is YYYYMMDD, unless otherwise specified.
- Any required formatting for a field is conveyed in the Bytes section. For example, Number(18,4) refers to number precision and scale. The first value is the precision and always matches the maximum number of digits for that field; the second value is the scale and specifies, of the total digits in the field, how many digits exist to the right of the decimal point. For example, the number –12345678901234.1234 would take up twenty ASCII characters in the flat file; however, the overall precision of the number is still (18,4).
- Field Order: Identifies the order of the field in the schema file.
- Required Field: Identifies whether the field can hold a null value. This section holds either a Yes or a No. A Yes signifies the field may not hold a null value. A No signifies that the field may, but is not required, to hold a null value.

General Business Rules and Standards Common to All APIs

- Complete snapshot (of what RDW refers to as dimension data):

  A majority of RDW dimension code requires a complete view of all current dimensional data (regardless of whether the dimension information has changed) once at the end of every business day. If a complete view of the dimensional data is not provided in the text file, invalid or incorrect dimensional data can result. For instance, not including an active item in the prditmdm.txt file causes that item to be closed (as of the extract date) in the data warehouse. When a sale for the item is processed, the fact program will not find a matching active dimension record. Therefore, it is essential, unless otherwise noted in each API's specific business rules section, that a complete snapshot of the dimensional data be provided in each text file.

  If there are no records for the day, an empty flat file must still be provided.

- Updated and new records of (what RDW refers to as fact data):

  Facts being loaded to RDW can either be new or updated facts. Unlike dimension snapshots, fact flat files will only contain new/updated facts exported from the source system once per day (or week, in some cases). Refer to each API's specific business rules section for more details.

  If there are no new or changed records for the day, an empty flat file must still be provided.

- Primary and local currency amount fields:

  Amounts will be stored in both primary and local currencies for most fact tables. If the source system uses multi-currency, then the primary currency column holds the primary currency amount, and the local currency column holds the local currency amount. If the location happens to use the primary currency, then both primary and local amounts hold the primary currency amount. If the source system does not use multi-currency, then only the primary currency fields are populated and the local fields hold NULL values.

- Leading/trailing values:

  Values entered into the text files are the exact values processed and loaded into the datamart tables. Therefore, the values with leading and/or trailing zeros, characters, or nulls are processed as such. RDW does not strip any of these leading or trailing values, unless otherwise noted in the individual API's business rules section.

- Indicator columns:

  Indicator columns are assumed to hold one of two values, either "Y" for yes or "N" for no.

- Delimiters:

  > **Note:** Make sure the delimiter is never part of your data.

  - Dimension Flat File Delimiter Standards (as defined by RDW): Within dimension text files, each field must be separated by a pipe (|) character, for example a record from prddivdm.txt may look like the following:

    ```
    1000|1|Homewares|2006|Henry Stubbs|2302|Craig Swanson
    ```

  - Fact Flat File Delimiter Standards (as defined by RDW): Within facts text files, each field must be separated by a semi-colon character (;). For example a record from exchngratedm.txt may look like the following:

    ```
    WIS;20010311;1.73527820592648544918
    ```

  See the latest RETL Programmer's Guide for additional information.

- ▪ End of Record Carriage Return:

    Each record in the text file must be separated by an end of line carriage return. For example, the three records below, in which each record holds four values, should be entered as:

    ```
    1|2|3|4
    5|6|7|8
    9|10|11|12
    ```

    and not as a continuous string of data, such as:

    ```
    1|2|3|4|5|6|7|8|9|10|11|12
    ```

sincilddm.txt

Business rules:

- ▪ This interface file contains invoice and order cost information for each item on a matched invoice.
- ▪ This interface file cannot contain duplicate transactions for an item_idnt, po_idnt, invc_idnt, supp_idnt, day_dt, and loc_idnt combination.
- ▪ This interface file follows the fact flat file interface layout standard.
- ▪ This interface file contains neither break-to-sell items nor packs that contain break-to-sell component items.

Name	Description	Data Type/Bytes	Field Order	Required Field
ITEM_IDNT	The unique identifier of an item.	CHARACTER(25)	1	Yes
PO_IDNT	The unique identifier of a purchase order.	VARCHAR2(8)	2	Yes
INVC_IDNT	The unique identifier of an invoice.	VARCHAR2(10)	3	Yes
SUPP_IDNT	The unique identifier of a supplier.	CHARACTER(10)	4	Yes
DAY_DT	The calendar day on which the transaction occurred.	DATE	5	Yes
LOC_IDNT	The unique identifier of the location.	CHARACTER(10)	6	Yes
F_SUPP_INVC_UNIT_COST_AMT	The invoice cost, in the system primary currency.	NUMBER(18,4)	7	No
F_SUPP_INVC_UNIT_COST_AMT_LCL	The invoice cost, in the local currency.	NUMBER(18,4)	8	No
F_SUPP_INVC_QTY	The quantity of an item shown on the invoice.	NUMBER(12,4)	9	No
F_PO_ITEM_UNIT_COST_AMT	The purchase order unit cost of the item, in primary currency.	NUMBER(18,4)	10	No
F_PO_ITEM_UNIT_COST_AMT_LCL	The purchase order unit cost of the item, in local currency.	NUMBER(18,4)	11	No