Oracle® Retail Store Inventory Management

Operations Guide

Release 14.1

E52928-02

January 2017

ORACLE®

Oracle Retail Store Inventory Management Operations Guide, Release 14.1

E52928-02

Primary Author: Tracy Gunston,

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

4 Batch Processes

5 Data Purge

List of Figures

List of Tables

Send Us Your Comments

Oracle Retail Store Inventory Management Operations Guide, Release 14.1

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

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

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

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

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

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

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

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

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

Preface

This operations guide provides critical information about the processing and operating details of Oracle Retail Store Inventory Management, including the following:

- System configuration settings
- Technical architecture
- Batch processing
- Data Purge

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementers
- Business analysts who need information about Oracle Retail Store Inventory Management processes and interfaces

Related Documents

For more information, see the following documents in the Oracle Retail Store Inventory Management Release 14.1 documentation set:

- *Oracle Retail Store Inventory Management Release Notes*
- *Oracle Retail Store Inventory Management Implementation Guide - Volume 1 - Configuration*
- *Oracle Retail Store Inventory Management Implementation Guide - Volume 2 - Integration with Oracle Retail Applications*
- *Oracle Retail Store Inventory Management Implementation Guide - Volume 3 - Mobile Store Inventory Management*
- *Oracle Retail Store Inventory Management Implementation Guide - Volume 4 - Extension Solutions*
- *Oracle Retail Store Inventory Management Installation Guide*
- *Oracle Retail Store Inventory Management User Guide*
- *Oracle Retail Store Inventory Management Data Model*

Documentation Accessibility

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

Access to Oracle Support

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

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Improved Process for Oracle Retail Documentation Corrections

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

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

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

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

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

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

Conventions

The following text conventions are used in this document:

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

1
Introduction

This operations guide serves as an Oracle Retail Store Inventory Management (SIM) reference to explain backend processes.

SIM empowers store personnel to sell, service, and personalize customer interactions by providing users the ability to perform typical back office functionality on the store sales floor. The results are greatly enhanced customer conversion rates, improved customer service, lower inventory carrying costs, and fewer markdowns. SIM delivers the information and flexible capabilities that store employees need to maintain optimal inventory levels and to convert shoppers into buyers.

The SIM solution does the following:

- Improves perpetual inventory levels by enabling floor-based inventory management through handheld devices and store PCs.
- Minimizes the time to process receipt and check-in of incoming merchandise.
- Receives, tracks, and transfers merchandise accurately, efficiently, and easily.
- Reduces technology costs by centralizing hardware requirements.
- Guides users through required transactions.
- Allows customizations to the product through an extensible technology platform. The retailer's modifications are isolated during product upgrades, lowering the total cost of ownership.

SIM is designed as a standalone application that can be customized to work with any merchandising system.

2

Technical Architecture

This chapter describes the overall software architecture for SIM, offering a high-level discussion of the general structure of the system.

SIM Technology Stack

SIM has an n-tier architecture consisting of a client tier, a server tier, and a data tier. The client tier contains a PC client (a Java desktop application) and handheld devices. The server tier contains the SIM server (deployed as a J2EE application inside the Weblogic Application Server) and the Wavelink server (a standalone server for the handheld devices). The data tier consists of an Oracle 11g database and an LDAP directory.

Advantages of the Architecture

SIM's robust distributed computing platform enables enhanced performance and allows for scalability.

The n-tier architecture of SIM allows for the encapsulation of business logic, shielding the client from the complexity of the backend system. Any given tier need not be concerned with the internal functional tasks of any other tier.

The following list is a summary of the advantages that accompany SIM's use of an n-tier architectural design:

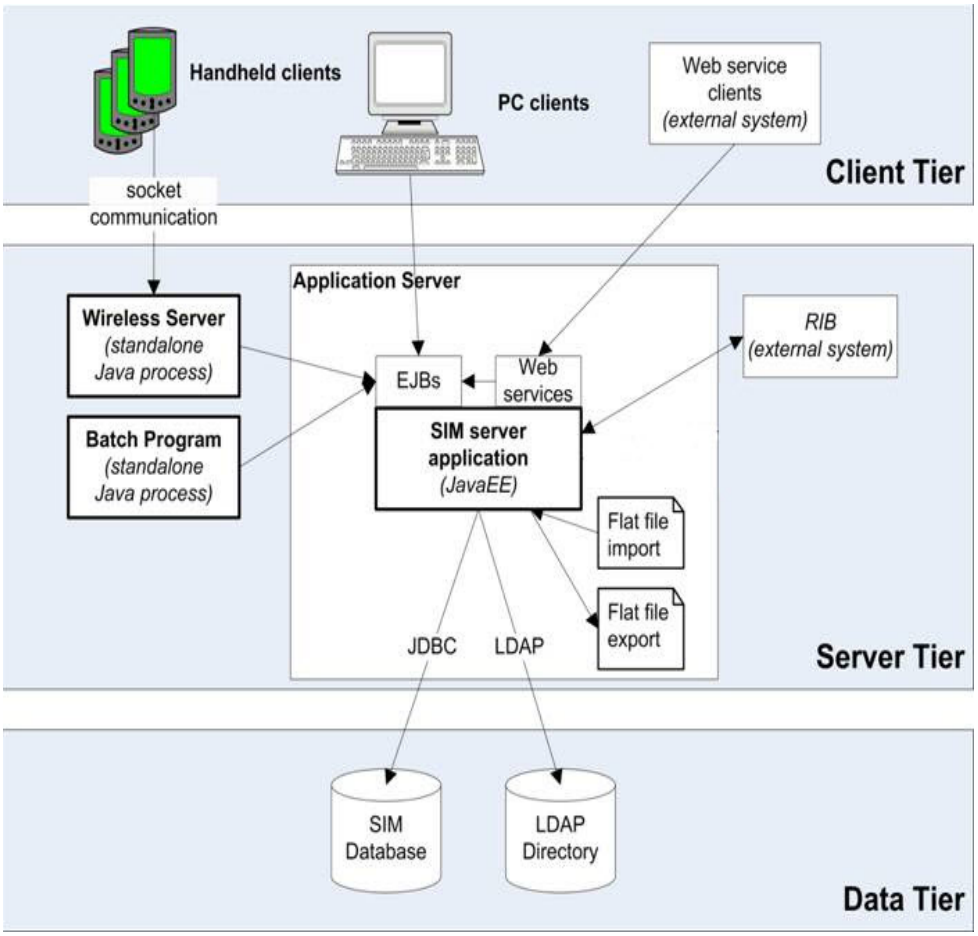
- Scalability: Hardware and software can be added to meet retailer requirements for each of the tiers.
- Maintainability: The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
- Platform independence: The code is written once but can run anywhere that Java can run.
- Cost effectiveness: Open source market-proven technology is utilized, while object-oriented design increases reusability for faster development and deployment.
- Ease of integration: The reuse of business objects and function allows for faster integration to enterprise subsystems. N-tier architecture has become an industry standard.
- High availability: Middleware is designed to run in a clustered environment or on a low-cost blade server.

- Endurance: Multi-tiered physically distributed architecture extends the life of the system.
- Flexibility: The system allocates resources dynamically based on the workload.

SIM Technical Architecture Diagrams and Description

This section provides a high-level overview of SIM's technical architecture. Figure 2–1 illustrates the major pieces of the typical three-tiered SIM implementation. Descriptions follow the diagram.

Figure 2–1 Three-Tiered SIM Implementation



Client Tier

SIM can be deployed on a wide variety of clients, including a desktop computer, a hand-held wireless device, and so on. The presentation tier only interacts with the middle tier (as opposed to the database tier). To optimize performance, the SIM PC front end facilitates robust client-side processing.

The PC side of SIM is built upon a fat client architecture, which was developed using Swing, a toolkit for creating rich GUIs in Java applications.

The handheld communication infrastructure piece, known as the Oracle Retail Wireless Foundation Server, enables the handheld devices to communicate with the

SIM server. The handheld devices talk to the Oracle Retail Wireless Foundation Server, which in turn makes calls as a client to the SIM server.

Note: SIM only exposes SOAP web services, a Web-services-client is NOT provided with the product. Customers can use their own clients to call SIM web services.

Middle (Server) Tier

By providing the link between the SIM client and the database, the middle tier handles virtually all of the business logic processing that occurs within SIM's multi-tiered architecture. The middle tier is comprised of services, most of which are related to business functionality. For example, an item service gets items, and so on. Within SIM, business objects are classes (that is, Java classes that have one or more attributes and corresponding set/get methods) that represent a functional entity.

Although the PC client and the handheld client use the middle tier's functionality differently, the middle tier is the same for both clients. The middle tier is designed to operate in a stateless manner, meaning it receives whatever instruction it needs to access the database from the client and does not retain any information between client calls.

If the workload warrants, SIM can be vertically scaled by adding additional application servers. Because SIM servers are running on multiple application servers in a stateless system, work can be seamlessly distributed among the servers.

SIM application servers can contain multiple containers, each of which is related to a unique Java Virtual Machine (JVM). Each container corresponds to a specific SIM instance. Introducing multiple instances of a container allows SIM retailers to more effectively distribute the processing among several containers and thereby horizontally scale the platform. As the request load for a service increases, additional instances of the service are automatically created to handle the increased workload.

The middle tier consists of the following core components, which allow it to make efficient and reliable calls to the SIM database:

- Server services contain the pertinent business logic.
- DAO classes handle database interaction.
- Databeans contain the SQL necessary to retrieve data from and save data to the database.

Note: There is at least one databean for every table and view in the database, but there may be more, used for different specific purposes.

Data Access Objects (DAO)

DAOs are classes that contain the logic necessary to find and persist data. They are used by services when database interaction is required.

Java Database Connectivity (JDBC)

DAOs communicate with the database through the industry standard Java database connectivity (JDBC) protocol. In order for the SIM client to retrieve the desired data from the database, a JDBC connection must exist between the middle tier and the database. JDBC facilitates the communication between a Java application and a

relational database. In essence, JDBC is a set of Application Programming Interfaces (APIs) that offer a database-independent means of extracting and/or inserting data to or from a database. To perform those insertions and extractions, SQL code also resides in this tier facilitating create, read, update, and delete actions.

Database Tier

The database tier is the application's storage platform, containing the physical data used throughout the application. The database houses data in tables and views; the data is used by the SIM server and then passed to the client. The database also houses stored procedures to do data manipulation in the database itself.

Distributed Topology

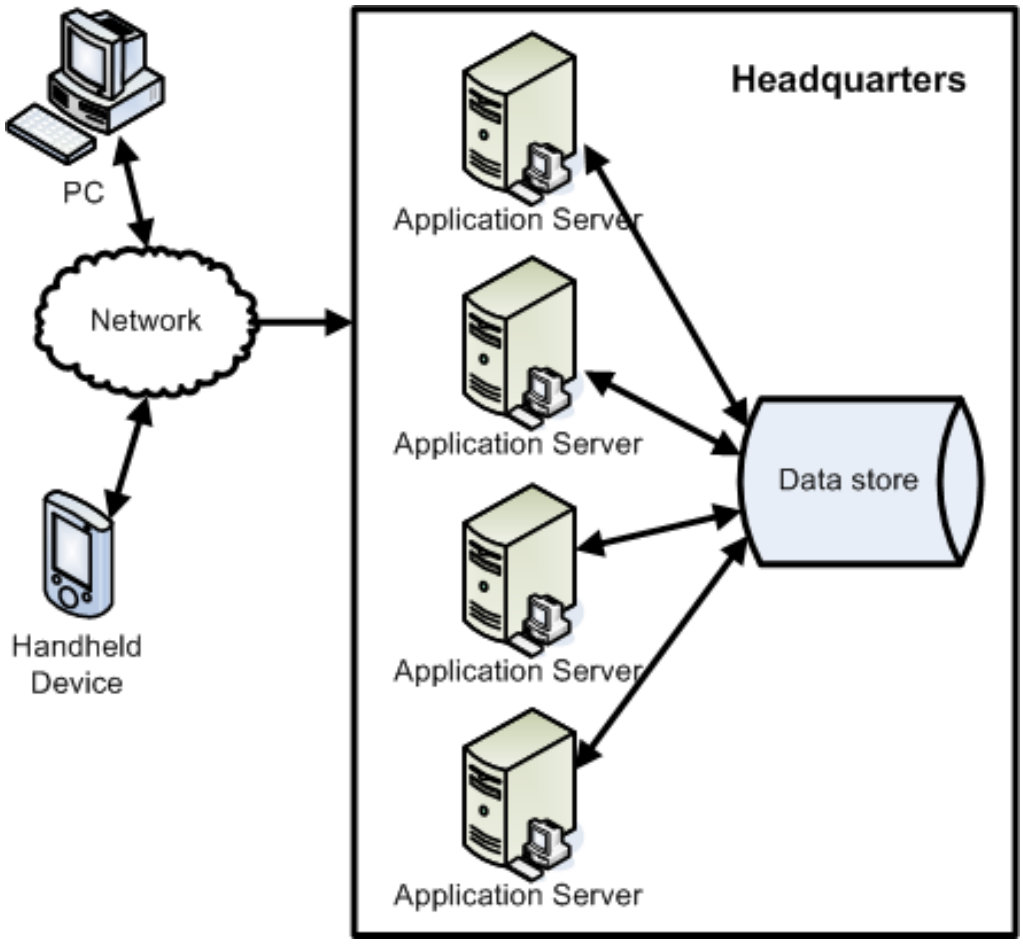
One of SIM's most significant advantages is its flexible distributed topology. SIM offers complete location transparency because the location of data and/or services is based upon the retailer's business requirements, not upon technical limitations. SIM's client server communication is an EJB call. Because the server does not have to be in the same store as the in-store clients, the clients log onto the server over the wire.

SIM's client code makes use of helper and framework classes that contain the logic to look up remote references to EJBs on the server and make calls to them. These helper and framework contain no business logic but contain only enough code to communicate with the server.

For example, if a helper class is called by the client to perform the method update shipment, the helper class appears to have that capability, though in reality it only behaves as a passage to the EJB remote reference, which is looked up from the server. The EJB remote reference communicates across the network with the server to complete the business-logic driven processing. The server performs the actual update shipment business logic and returns any return values or errors to the client.

Connectivity between the SIM client and the middle tier is achieved through the Java Naming and Directory Interface (JNDI), which the SIM client accesses with the necessary IP address and port. JNDI contains the means for the client to look up services available on the application server.

Figure 2–2 SIM Deployments



3

Backend System 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 key system parameters, logging settings, and exception handling.

Supported Oracle Retail Products/Environments

For information about integration compatibility for this release and for requirements for SIM's client, servers, and database, see the *Oracle Retail Store Inventory Management Installation Guide*.

Configuring SIM Across Time Zones

For many SIM retailers, a corporate server is located in a different time zone than the stores connected to that corporate server. When a transaction is processed at these respective locations, there is timestamp information associated with these transactions. SIM has the ability to reconcile these time zone differences.

Setup RIB Interface GMT Parameter

System administration options enable you to specify the time zone to use when timestamps are published to or received from the Oracle Retail Integration Bus (RIB).

For detailed descriptions of list of GMT Parameter, see the *Oracle Retail Store Inventory Management Implementation Guide*, *Volume 1*, section "Setup and Configuration".

Note: When Enable GMT is set to "Yes", the dates published to the RIB will be in GMT, and incoming timestamps in RIB messages will also be read as GMT; When the value is set to "No", timestamps are published to the RIB in the store time zone, and incoming timestamps in RIB messages will be read as the store time zone.

When integrate with systems which dates are not in GMT or not time zone specific, then the enable GMT configuration must be set to "No".

GUI based System and Store Configuration

SIM provides GUI screens for administrators to set some of the system and store configuration settings. For details, see the *Oracle Retail Store Inventory Management Implementation Gui*de, section "System and Store Administration".

Configuration Files

Key system configuration parameters are described in this section. Key client-defined configurations for SIM are described in this section. The system parameters contained in these files are also detailed. Many parameters have been omitted from this section because retailers should not have to change them. Most of the configuration settings are set by SIM installer when retailers install SIM.

SIM configuration files are packaged as various resources jars, this section list the most commonly used configuration files.

Some settings in the files are configurable. Thus, when retailers install SIM into an environment, they must update these values to their specific settings.

Note: When manually making configuration file changes, the updated .ear files must be redeployed.

SIM JNDI Configuration

The following section describes the SIM JNDI configuration.

jndi.cfg

The JNDI property file allows you to specify the properties to be used during client (SIM PC client, SIM wireless client, SIM batch client) looks up SIM server. The jndi.cfg is packaged in sim-client-resources.jar, sim-batch.zip, and sim-wireless.zip distributions.

Example 3–1 jndi.cfg File

```
# The InitialContextFactory class (Required)
INITIAL_CONTEXT_FACTORY=weblogic.jndi.WLInitialContextFactory

# The URL for the naming server (Required)
NAMING_SERVER_URL=@jndi.naming.server.url@

# The implementation used for providing security credentials (Required)
SECURITY_CREDENTIAL_
PROVIDER=oracle.retail.sim.common.security.CredentialStoreJndiCredentialProvider

# Credential provider parameters (Depends on credential provider)
SECURITY_USER_ALIAS=batch-user

# The maximum number of times to try to connect to the server.
# Should always be at least 2 to allow for server unexpected reconnects.
MAX_CONNECT_ATTEMPTS=2
```

Note: Within configuration files (and, thus, in some of the examples from those following files), a # sign that precedes a value in the file signifies that what follows is a comment and is not being utilized as a setting.

Example 3–2 web-launch.properties

```
# SIM Web Launch Configuration

initializers=oracle.retail.sim.weblaunch.bootstrap.WebLaunchInitializer
```

```
finalizers=

context.handler=oracle.retail.sim.weblaunch.core.DefaultWebLaunchContextHandler

token.codebase_url=@deploy.client.codebase.url@
token.jndi_url=@jndi.naming.server.url@

security.sso.enabled=@security.sso.enabled@
security.sso.header.user=OAM_REMOTE_USER
security.sso.token.expiration=3600
security.sso.token.encryption.provider=oracle.retail.sim.weblaunch.security.SimTok
enEncryptionProvider
security.sso.token.key.generate=true
security.sso.token.key.alias=sso-token-key
security.sso.token.key.algorithm=HmacSHA256
security.sso.token.random.algorithm=SHA1PRNG
security.sso.token.cipher.algorithm=AES/CBC/PKCS5Padding
security.sso.token.cipher.key.size=128
security.sso.token.cipher.iv.length=16
security.sso.token.mac.algorithm=HmacSHA256
security.sso.token.mac.key.size=256
security.sso.token.mac.salt.length=32
```

Batch Configuration

Batch configuration files contains configuration settings for SIM batch programs. The batch configuration files are packaged in sim-batch.zip distribution.

batch.cfg

The batch.cfg file contains the following:

Example 3–3 batch.cfg File

```
# The number of threads that execute concurrently to get batch work done when
#SimBatch.executeBatchCallables is used.
BATCH_NUM_THREADS_IN_POOL=5

# Time in seconds to wait when checking the status of an asynchronous batch job
BATCH_ASYNC_DELAY=5

# The maximum number of times to check the status of an asynchronous batch job
BATCH_ASYNC_LIMIT=50
```

jndi.cfg

For details, see the section "SIM JNDI Configuration."

Jps-config.xml

This file contains identity store provider configuration.

For details, see the *Oracle Retail Store Inventory Management Implementation Guide*, section "Oracle Software Security Assurance (OSSA)".

Log4j.xml

This file is used for changing the logging level for SIM batch programs. For details, see the section "Changing Logging Levels".

Common Configuration

The configuration files which are common to both client code and server code.

Common configuration files are packaged in sim-common-resources.jar distribution.

> **Note:** If a configuration file is changed, all client side jars containing Java code must be signed with the same signature.

common.cfg

The following keys define the implementation to classes that instantiate objects in SIM through the factory pattern:

- BO_FACTORY_IMPL
- WSO_FACTORY_IMPL
- DEO_FACTORY_IMPL
- CLIENT_COMMAND_FACTORY_IMPL
- CLIENT_SERVICE_FACTORY_IMPL
- SERVER_SERVICE_FACTORY_IMPL

Where:

- BO – Business Objects
- WSO – Web Service Objects
- DEO – Data Exchange Objects

The common.cfg file also includes server caching refresh rates for various services.

Example 3–4 common.cfg File

```
# These keys define the implementation to classes that instantiate objects in SIM
via the factory pattern
# BO = Business Objects, WSO = Web Service Objects, DEO = Data Exchange Objects
BO_FACTORY_IMPL=oracle.retail.sim.common.business.BOFactoryImpl
CLIENT_COMMAND_FACTORY_
IMPL=oracle.retail.sim.common.business.ClientCommandFactoryImpl
CLIENT_SERVICE_FACTORY_
IMPL=oracle.retail.sim.service.core.ClientServiceFactoryImpl
SERVER_SERVICE_FACTORY_
IMPL=oracle.retail.sim.service.core.ServerServiceFactoryImpl

# Default currency type for non-specified currency
CURRENCY_DEFAULT_TYPE=USD

# Indicates the maximum number of lines that will be handled by authorization
process at one time.
# In other words, authorization of stock count will loop on total line items and
only process 5000
# line items within each loop.
STOCK_COUNT_MAX_AUTH_LINES=5000

# These mask factories determine the implementation responsible for instantiating
the money, phone masks (parsing/formatting).
MONEY_MASK_FACTORY=oracle.retail.sim.common.format.MoneyMaskFactoryImpl
PHONE_MASK_FACTORY=oracle.retail.sim.common.format.PhoneMaskFactoryImpl
```

```
# The implementation of the credential store provider
CREDENTIAL_STORE_
PROVIDER=oracle.retail.sim.common.security.JpsCredentialStoreProvider

# Map name used for accessing the credential store
CREDENTIAL_STORE_MAP=oracle.retail.sim

# Cache Refresh Rates Of Commonly Cached Information.
# Value is in milliseconds. 30000=30 seconds  300000=5 minutes  3600000=1 hour
# The default setting is 1 hour unless configured here

# Reason Codes for inventory adjustments cached on handheld server and PC Client
(milliseconds)
REFRESH_RATE_ACTIVE_INV_ADJ_REASON=3600000

# System configuration values cached on handheld server, main server and PC client
(milliseconds)
REFRESH_RATE_CONFIG=3600000

# Context Types cached on handheld server, main server and PC client
(milliseconds)
REFRESH_RATE_CONTEXT_TYPE=3600000

# Return reason code  for a return to finisher cached on the handheld server and
PC client (milliseconds)
REFRESH_RATE_FINISHER_RETURN_REASON=3600000

# AGSN Item Ticket formats cached on the handheld server and PC client
(milliseconds)
REFRESH_RATE_AGSN_TICKET_FORMAT=3600000

# Regular Item Ticket Formats cached on the handheld server and PC client
(milliseconds)
REFRESH_RATE_ITEM_TICKET_FORMAT=3600000

# Regular Item Ticket Types cached on the handheld server and PC client
(milliseconds)
REFRESH_RATE_ITEM_TICKET_TYPE=3600000

# Item merchandise hierarchy (depts) cached on handheld server, main server and PC
client (milliseconds)
REFRESH_RATE_MERCH_HIERARCHY=3600000

# Non-Sellable Quantity Types or Sub-buckets cached on handheld server, main
server and PC client (milliseconds)
REFRESH_RATE_NONSELLABLE_QTY_TYPE=3600000

# Price history cached on handheld server and PC client (milliseconds)
REFRESH_RATE_PRICE_HISTORY=3600000

# Shelf label formats cached on the handheld server and PC client (milliseconds)
REFRESH_RATE_SHELF_LABEL_FORMAT=3600000

# Store configuration cached on handheld server, main server and PC client
(milliseconds)
REFRESH_RATE_STORE=3600000

# Suppliers cached on handheld server and PC client (milliseconds)
REFRESH_RATE_SUPPLIER=3600000
```

```
# Return reason codes for supplier returns on handheld server and PC client
(milliseconds)
REFRESH_RATE_SUPPLIER_RETURN_REASON=3600000

# Translations cached on handheld server and main server (milliseconds)
REFRESH_RATE_TRANSLATION=3600000

# UDA Details cached on PC client (milliseconds)
REFRESH_RATE_UDA_DETAILS=3600000

# UIN Container details cached on handheld server (milliseconds)
REFRESH_RATE_UIN_CONTAINER=3600000

# UOM Conversion Factor cached on handheld server and main server (milliseconds)
REFRESH_RATE_UOM_CONVERSION=3600000

# Warehouses cached on handheld server and PC client (milliseconds)
REFRESH_RATE_WAREHOUSE=3600000

# Return reason codes for warehouse returns on handheld server and PC client
(milliseconds)
REFRESH_RATE_WAREHOUSE_RETURN_REASON=3600000

# Differentiator information cached on handheld server (milliseconds)
REFRESH_RATE_WIRELESS_ITEM_DIFF=3600000
```

PC Client Configuration

Client configuration files are package in sim-client-resources.jar distribution.

> **Note:** If a configuration file is changed, all client side jars containing Java code must be signed with the same signature.

client.cfg

The client.cfg file contains the following:

- STARTUP_DISPLAY – this class must be StartupDisplayer or a sub-class of StartupDisplayer.
- INITIALIZERS – comma-delimited class name list that is executed upon SIM PC client startup.
- NATIVE.COMMANDS – comma-delimited list of native commands that display in stats area.
- CLIENT_LOCK_PORT – the port used to lock the client application.
- SHUTDOWN – shutdown command line.
- Screen components used by application main frame:
  - GUI.APPFOLDER
  - GUI.MAINFRAME
  - GUI.GLOBALBAR
  - GUI.APPTOOLBAR
  - GUI.LOGGERFACTORY

  - GUI.STATUSDISPLAYER
  - GUI.DEFAULT_SCREEN
  - GUI.EXIT
    * NORMAL
    * SPIN
    * DISSOLVE
    * SHRINK
- REPORTING_SERVICE_BROWSER_LAUNCHER –Reports Browser launcher.
- REPORTS_EXECUTABLE – this value holds the executable for displaying the reports portal.
- HELP_EXECUTABLE – this value holds the executable for displaying help.

Example 3–5 client.cfg File

```
# This class must be StartupDisplayer or a sub-class of StartupDisplayer
STARTUP_DISPLAY=oracle.retail.sim.closed.application.StartupDisplayer

# A comma delimited class name list that is executed upon SIM PC client startup
# Each entry must be an implementation of
oracle.retail.sim.closed.common.Initializer.
INITIALIZERS=oracle.retail.sim.closed.bootstrap.SimClientInitializer

# Comma delimited list of native commands that will show up in stats area.
NATIVE.COMMANDS=

# The port which is used to lock the client application.  This port will be
checked on each client invocation to make sure only one instance of this
application is running.
CLIENT_LOCK_PORT=51803

# Shutdown command line
SHUTDOWN=RESTART_CLIENT

# Screen components used by application main frame
GUI.APPFOLDER=sim
GUI.MAIN_FRAME=oracle.retail.sim.closed.swing.sim.SimApplicationFrame
GUI.GLOBALBAR=oracle.retail.sim.shared.swing.core.SimStatusBar
GUI.APPTOOLBAR=oracle.retail.sim.shared.swing.core.SimToolbar
GUI.LOGGER_FACTORY=oracle.retail.sim.shared.swing.core.SimLoggerFactory
GUI.STATUS_DISPLAYER=oracle.retail.sim.shared.swing.core.SimStatusDisplayer
GUI.DEFAULT_SCREEN=oracle.retail.sim.shared.swing.login.MainScreen
GUI.WRAPPER_FACTORY=oracle.retail.sim.client.core.ClientWrapperFactoryImpl
GUI.NAV_LISTENER_
FACTORY=oracle.retail.sim.client.core.SimScreenNavigationListenerFactoryImpl

# Screen Exit Options are NORMAL, SPIN, DISSOLVE, SHRINK
GUI.EXIT=NORMAL

# Reports Browser launcher
REPORTING_SERVICE_BROWSER_
LAUNCHER=oracle.retail.sim.shared.report.launcher.bipublisher.BIPublisherBrowserRe
portLauncher

# This value holds the executable for displaying the reports portal.
# Need a different executable for Linux
```

```
REPORTS_EXECUTABLE=cmd /c start {0}

# This value holds the executable for displaying help.
# Available parameters: {0} - the absolute path of the URL which should display
#                             the help documentation.
HELP_EXECUTABLE=cmd /c start {0}
```

date.cfg

This file defines the date format configuration.

This file contains Java format pattern strings for several different types of dates defined in the system. These pattern strings follow the rules defined in Java for SimpleDateFormat. The key for the date is defined as language and country followed by the pattern key where xxXX is the two-letter ISO language code plus country code. Both language and country must be present. Additional language/country combinations can be added as desired. For example, enAU.entryDate is the entry format for dates in English for Australia.

The pattern keys are:

- entryDate–used for date entry in calendar editor
- shortDate–format for short length date - this is the most commonly used
- mediumDate–format for medium length date
- longDate–nearly complete date format
- fullDate–fully written-out date format
- monthPattern–formats month and day only
- wirelessInput–defines entry for wireless device
- wirelessOutput–defines the format of dates on the wireless device
- wirelessDisplay–defines the exact text string to display to the user at the entry location
- firstDayOfWeek – the first day of the week to display on calendar pop-up

Editing date.cfg

The file must be changed and then placed in the classpath.

Valid days of week values are:

- 1 = Sunday
- 2 = Monday
- 3 = Tuesday
- 4 = Wednesday
- 5 = Thursday
- 6 = Friday
- 7 = Saturday

The enUS represents the language (en=english) and country (US=United States) of the user logged in. There are examples for many other countries already provided in the date.cfg file.

Example 3–6 date.cfg File

```
# Date formats are in standard JAVA pattern definition strings
# First Day of Week: 1 = Sunday....7 = Saturday

# ENGLISH - UNITED STATES
#enUS.firstDayOfWeek=1
#enUS.entryDate=M/d/yy
enUS.shortDate=M/d/yyyy
#enUS.mediumDate=MMM d, yyyy
#enUS.longDate=MMMM d, yyyy
#enUS.fullDate=EEEE, MMMM d, yyyy
enUS.monthPattern=MM-dd
enUS.wirelessInput=MM-dd-yy,MMddyy
enUS.wirelessOutput=MM-dd-yy
enUS.wirelessDisplay=mm-dd-yy

# FRENCH - BELGIUM
#frBE.firstDayOfWeek=1
#frBE.entryDate=d/MM/yy
#frBE.shortDate=d/MM/yy
#frBE.mediumDate=dd-MMM-yyyy
#frBE.longDate=d MMMM yyyy
#frBE.fullDate=EEEE d MMMM yyyy
frBE.monthPattern=MM-dd
frBE.wirelessInput=dd-MM-yy,ddMMyy
frBE.wirelessOutput=dd-MM-yy
frBE.wirelessDisplay=dd-mm-yy
```

jdni.cfg

For details, see section "SIM JNDI Configuration."

Log4j.xml

This file is used for changing the logging level for SIM client programs. For details, see section "Changing Logging Levels."

Server Configuration

SIM server configuration files are packaged in sim-server-resources.jar distribution.

ldap.cfg

This file contains various configuration parameters for connecting to an LDAP server. The SIM installer should have set all values.

Additional context properties (specific to the context factory implementation) may be set if needed, see the comments for examples.

The configuration allows for customizations of the LDAP schema for object context, object class names, attribute names.

Example 3–7 ldap.cfg File

```
# LDAP Configuration

# Connection factory implementation
connection.factory=oracle.retail.sim.server.dataaccess.LdapThreadLocalConnectionFa
ctory
```

```
# Initial context factory implementation (java.naming.factory.initial)
initial.context.factory=com.sun.jndi.ldap.LdapCtxFactory

# Provider URL (java.naming.provider.url)
provider.url.primary=@ldap.url@
provider.url.backup=


# Security authentication (java.naming.security.authentication)
security.authentication=simple

# Security protocol (java.naming.security.protocol)
security.protocol=

# Security credentials
security.user.alias=ldap-user

# Connection pool (com.sun.jndi.ldap.connect.pool)
connection.pool=true

# Additional context environment properties (context.env.*)
#context.env.com.sun.jndi.ldap.connect.timeout=
#context.env.com.sun.jndi.ldap.read.timeout=

# Base DN
base.dn=@ldap.base.dn@

# Store schema
schema.store.context=cn=SIMStores
schema.store.objectclass=simStore
schema.store.attribute.store.id=storeId

# Role schema
schema.role.context=cn=SIMRoles
schema.role.objectclass=simRole
schema.role.attribute.role.name=roleName
schema.role.attribute.type=type
schema.role.attribute.description=description

# User schema
schema.user.context=cn=Users
schema.user.objectclass=simUser
schema.user.attribute.username=uid
schema.user.attribute.superuser=superUser
schema.user.attribute.status=empStatus
schema.user.attribute.language=preferredLanguage
schema.user.attribute.country=preferredCountry
schema.user.attribute.first.name=givenName
schema.user.attribute.middle.name=middleName
schema.user.attribute.last.name=sn
schema.user.attribute.email=mail
schema.user.attribute.phone=telephoneNumber
schema.user.attribute.external.id=externalId
schema.user.attribute.supervisor=supervisor
schema.user.attribute.comments=description
schema.user.attribute.create.date=createTimestamp
schema.user.attribute.start.date=startTimestamp
schema.user.attribute.end.date=endTimestamp
schema.user.attribute.default.store=defaultStore
```

```
schema.user.attribute.user.stores=userStores

# User store schema
schema.user.store.context=cn=Users
schema.user.store.objectclass=simUser
schema.user.store.attribute.username=uid
schema.user.store.attribute.user.stores=userStores

# User role schema
schema.user.role.context=cn=Users
schema.user.role.objectclass=simUserRole
schema.user.role.attribute.role.name=roleName
schema.user.role.attribute.user.role=userRole
schema.user.role.attribute.user.role.stores=userRoleStores
schema.user.role.attribute.start.date=startTimestamp
schema.user.role.attribute.end.date=endTimestamp
```

server.cfg

The server.cfg file contains server side configurations. For details, see the example server.cfg file.

Example 3–8 server.cfg File

```
# INITIALIZERS: A comma delimited class name list that needs to be executed when
the #sim server starts.
# Each entry must implement oracle.retail.sim.common.core.Initializer.
INITIALIZERS=oracle.retail.sim.server.bootstrap.SimServerInitializer,oracle.retail
.sim.server.bootstrap.IntegrationServiceServerInitializer,oracle.retail.sim.intrib
.bootstrap.IntegrationRibServerInitializer

# FINALIZERS: A comma delimited class name list that needs to be executed when the
sim #server stops.

# FINALIZERS: A comma delimited class name list that needs to be executed when the
sim #server stops.
# Each entry must implement oracle.retail.sim.common.Finalizer.
# For an example of what could be done, see oracle.retail.sim.tests.TestFinalizer
FINALIZERS=oracle.retail.sim.server.bootstrap.SimServerFinalizer

# Enable when deployed in a clustered environment to allow for cluster specific
features.
CLUSTERED=@deploy.clustered@

# The name the application's DataSource is registered under.
DB_JNDI_NAME=jdbc/SimDataSource

# The name the application's MailSession is registered under.
MAIL_JNDI_NAME=mail/SimMailSession

# Time in seconds to wait on database locks
DB_LOCK_WAIT_TIME=60
DB_LOCK_WAIT_TIME_STOCK_COUNT=300

# Maximum number of parameters allowed in a database batch statement
DB_BATCH_MAX_PARAM=5000

# Default database fetch size for batch processes
BATCH_FETCH_LIMIT_DEFAULT=100
```

```
# These keys define the implementation to classes that instantiate objects in SIM
via #the factory pattern
DEO_FACTORY_IMPL=oracle.retail.sim.server.integration.deo.DEOFactoryImpl
DAO_FACTORY_IMPL=oracle.retail.sim.server.dataaccess.DAOFactoryImpl
EXTERNAL_SERVICE_FACTORY_
IMPL=oracle.retail.sim.extservice.core.ExternalServiceFactoryImpl
COMMAND_FACTORY_IMPL=oracle.retail.sim.server.business.CommandFactoryImpl
RECORD_FACTORY_IMPL=oracle.retail.sim.server.business.RecordFactoryImpl

# These factories lookup appropriate message types, mappers, consumers,
publishers, #and stagers for message processing
MESSAGE_ENUM_FACTORY_
IMPL=oracle.retail.sim.server.integration.SimMessageEnumFactoryImpl
MESSAGE_MAPPER_FACTORY_
IMPL=oracle.retail.sim.server.integration.SimMessageMapperFactoryImpl
MESSAGE_CONSUMER_FACTORY_
IMPL=oracle.retail.sim.server.integration.consumer.SimMessageConsumerFactoryImpl
MESSAGE_STAGER_FACTORY_
IMPL=oracle.retail.sim.server.integration.stager.SimMessageStagerFactoryImpl
MESSAGE_PUBLISHER_FACTORY_
IMPL=oracle.retail.sim.server.integration.publisher.SimMessagePublisherFactoryImpl

# Server cache refresh rates. Time is in milliseconds. 30000=30 seconds 300000=5
minutes
REFRESH_RATE_USER_AUTHENTICATION_CACHE=300000
REFRESH_RATE_USER_AUTHORIZATION_CACHE=300000

# The implementation of the internal security password encryption provider
INTERNAL_PASSWORD_ENCRYPTION_
PROVIDER=oracle.retail.sim.server.security.SimPasswordEncryptionProvider
# The algorithm name for random number generation used for internal security
INTERNAL_PASSWORD_RANDOM_ALGORITHM=SHA1PRNG
# The credential store alias for accessing the internal security password
encryption key
INTERNAL_PASSWORD_KEY_ALIAS=internal-password-key
# Enables initial automatic generation of the internal security password
encryption key
INTERNAL_PASSWORD_KEY_GENERATE=true
# The algorithm name for internal security password encryption key generation
INTERNAL_PASSWORD_KEY_ALGORITHM=HmacSHA256
# The MAC algorithm name used for internal security password encryption
INTERNAL_PASSWORD_MAC_ALGORITHM=HmacSHA256
# The size of the MAC encryption key used for internal security password
encryption (bits)
INTERNAL_PASSWORD_MAC_KEY_SIZE=256
# The length of the salt generated for internal security password encryption
(bytes)
INTERNAL_PASSWORD_MAC_SALT_LENGTH=32

# This corresponds to a row in the message processing system configuration table
MPS_CONFIG_ID=1
```

Wireless Configuration

SIM wireless configuration files are packaged in sim-wireless.zip distribution.

wireless.cfg

This file contains configuration used by the Wireless Server:

- INITIALIZERS_CLIENT – comma-delimited class name list that is executed upon wireless client startup.
- INITIALIZERS_SERVER – comma-delimited class name list that is executed upon wireless server startup.
- Wireless Port – port the wireless server runs on, and clients connect to.

Example 3–9 wireless.cfg File

```
# INITIALIZERS_CLIENT: A comma delimited class name list that is executed upon
wireless client startup
# Each entry must be an implementation of
oracle.retail.sim.common.core.Initializer.
INITIALIZERS_
CLIENT=oracle.retail.sim.wireless.bootstrap.SimWirelessClientInitializer

# INITIALIZERS_SERVER: A comma delimited class name list that is executed upon
wireless server startup
# Each entry must be an implementation of
oracle.retail.sim.common.core.Initializer.
INITIALIZERS_
SERVER=oracle.retail.sim.wireless.bootstrap.SimWirelessServerInitializer

# Wireless Port - This is the port that the wireless server runs on, and clients
connect to.
PORT=@wireless.port@
```

jndi.cfg

For details, see "SIM JNDI Configuration."

Jps-config.xml

This file contains identity store provider configuration.

For details, see the *Oracle Store Inventory Management Implementation Guide*, section "Oracle Software Security Assurance (OSSA)".

Log4j.xml

This file is used for changing the logging level for SIM wireless programs. For details, see the section "Changing Logging Levels".

External Service Integration Configuration

External services configuration files contain configurations for SIM to be able to send requests to external services. External service configuration files are packaged in sim-ext-services-resources.jar distribution.

ext-services.cfg

An example of ext-services.cfg is as follows:

Example 3–10 ext-services.cfg File

```
# RpmExternalServices
# BIPublisherSchedulerExternalServices
BIPublisherScheduleExternalServices.user.alias=bip-user
```

```
# BIPublisherExternalServices
BIPublisherExternalServices.wsdl.url=@integration.bip.wsdl.url@
BIPublisherExternalServices.user.alias=@integration.bip.user.alias@

# ManifestExternalServices
ManifestExternalServices.wsdl.url=@integration.manifest.wsdl.url@
ManifestExternalServices.decorator=@integration.manifest.decorator@
ManifestExternalServices.user.alias=manifest-user
ManifestExternalServices.client.keystore.alias=manifest-client-keystore@

ManifestExternalServices.client.key.alias=manifest-client-key
ManifestExternalServices.server.key.name=@integration.manifest.server.key.name@

# FulfillmentOrderExternalServices
FulfillmentOrderExternalServices.wsdl.url=@integration.oms.wsdl.url@
FulfillmentOrderExternalServices.decorator=@integration.oms.decorator@
FulfillmentOrderExternalServices.user.alias=oms-user

FulfillmentOrderExternalServices.client.keystore.alias=oms-client-keystore
FulfillmentOrderExternalServices.client.key.alias=oms-client-key
FulfillmentOrderExternalServices.server.key.name=@integration.oms.server.key.name@
```

service_flavors.xml

This file is one of the Retail Platform's service factory configuration files which specifies the service request context (or flavors), for example, from a client application or the request is originated from a process executing on the server.

The services_<application>.xml file (for example, services_ribclient.xml,) specifies instructions to the Retail Platform's service factory indicating which type of the service to use based on the context of the service request.

By default, this file should not be modified.

Example 3–11 service_flavors.xml

```
<?xml version="1.0" encoding="UTF-8"?>
  <services-config>
     <flavors set="client">
         <!-- Default flavor precedence and implementation -->
         <flavor name="ejb" locator="com.retek.platform.service.EjbServiceLocator"
remote-suffix="Remote" home-suffix="RemoteHome" />
        <!--flavor name="core"
locator="com.retek.platform.service.SimpleServiceLocator" suffix="Impl"/-->
         <flavor name="file"
locator="com.retek.platform.service.SimpleServiceLocator" prefix="File" />
         <flavor name="offline"
locator="com.retek.platform.service.SimpleServiceLocator" prefix="Offline" />
     </flavors>
     <flavors set="server">
         <!-- Default flavor precedence and implementation -->
        <!-- Default flavor precedence and implementation -->
         <flavor name="java"
locator="com.retek.platform.service.SimpleServiceLocator" suffix="Java" />
         <flavor name="ejblocal"
locator="com.retek.platform.service.EjbServiceLocator" remote-suffix="Local"
home-suffix="LocalHome" jndi-name-suffix="Local" />
         <flavor name="core"
locator="com.retek.platform.service.SimpleServiceLocator" suffix="Impl" />
```

```
     </flavors>
</services-config>
```

RIB Integration Configuration

RIB integration configuration files contain configurations for SIM integrates with Retail Products through Retail Integration Bus. RIB configuration files are packaged in sim-int-rib-resources.jar distribution.

remote_service_locator_info_ribclient.xml

This file contains the remote service lookup configuration to invoke RIB remote EJBs for publishing messages to RIB.

The SIM installer configures the configurable entries at SIM installation time. The SIM Installer also creates the security authentication wallet file and deploys to the SIM application server.

Example 3–12 remote_service_locator_info_ribclient.xml

```
<?xml version="1.0" ?>
<remote_service_locator_info>
<!-- This is for JNDI  -->
     <provider id="rib-sim" map-name="oracle.retail.sim"
csm-wallet="@security.credstore.path@" >
          <context-property name="java.naming.factory.initial"
value="weblogic.jndi.WLInitialContextFactory" />
          <context-property name="java.naming.provider.url"
value="@integration.rib.url@" />
       <encrypted-context-property name="java.naming.security.principal"
key="rib-user" value-type="PC_USERNAME" />
       <encrypted-context-property name="java.naming.security.credentials"
key="rib-user" value-type="PC_PASSWORD" />
     </provider>
  </remote_service_locator_info>


where
@security.credstore.path@:  The location where the security authentication wallet
file is located

@integration.rib.url@: The RIB-SIM server's JNDI URL.

For example:
t3://rib-sim-serverxxxx:19106
Or
t3://rib-sim-serverxxxx:19106/rib-sim
```

Note: See the *Oracle Retail Store Inventory Management Installation Guide* for wallet location and configuration details.

retail_service_config_info_ribclient.xml

This file is an RIB integration-related service factory configuration file. It specifies the RIB application's interfaces and their associated implementations.

By default, this file should not be modified.

Example 3–13 retail_service_config_info_ribclient.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<retail_service_config_info>
<!-- POJO services listing -->
<pojo-service-config>
<interface package="com.retek.rib.binding.publisher">
<impl  package="com.retek.rib.binding.publisher.impl" suffix="Impl" />
</interface>
</pojo-service-config>
<!-- EJB services listing -->
<remote-ejb-service-config>
<interface package="com.retek.rib.app.messaging.publisher.service">
<impl package="com.retek.rib.app.messaging.publisher.service.impl"
remote-bean-prefix="" remote-bean-suffix="Remote" remote-home-prefix=""
remote-home-suffix="RemoteHome" remote-service-locator-info-ref-id="rib-sim" />
 </interface>
 </remote-ejb-service-config>
</retail_service_config_info>
```

injectors.xml

This file contains mappings for RIB message injector handler class and RIB incoming message family/type.

By default, this file should not be modified.

Example 3–14 injectors.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<injector_config>
<family name="asnin">
<injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
<type>asnincre</type>
<type>asnindel</type>
<type>asninmod</type>
</injector>
</family>
<family name="diffs">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>diffcre</type>
            <type>diffdel</type>
            <type>diffmod</type>
        </injector>
    </family>
<family name="fulfilord"
drop-messages-of-types="fulfilordpocre,fulfilordtsfcre,fulfilordapprdel">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>fulfilordstdlvcre</type>
            <type>fulfilordreqdel</type>
        </injector>
    </family>

    <family name="items">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>itembomcre</type>
            <type>itembomdel</type>
            <type>itembommod</type>
            <type>itemcre</type>
```

```
            <type>itemdel</type>
            <type>itemhdrmod</type>
            <type>itemtcktcre</type>
            <type>itemtcktdel</type>
            <type>itemtcktmod</type>
            <type>itemsupctycre</type>
            <type>itemsupctydel</type>
            <type>itemsupctymod</type>
            <type>itemsupcre</type>
            <type>itemsupdel</type>
            <type>itemsupmod</type>
            <type>itemupccre</type>
            <type>itemupcdel</type>
            <type>itemupcmod</type>
            <type>iscmfrcre</type>
            <type>iscmfrdel</type>
            <type>iscmfrmod</type>
            <type>iscdimcre</type>
            <type>iscdimdel</type>
            <type>iscdimmod</type>
            <type>itemimagecre</type>
            <type>itemimagemod</type>
            <type>itemimagedel</type>
            <type>itemudadatecre</type>
            <type>itemudadatedel</type>
            <type>itemudadatemod</type>
            <type>itemudaffcre</type>
            <type>itemudaffdel</type>
            <type>itemudaffmod</type>
            <type>itemudalovcre</type>
            <type>itemudalovdel</type>
            <type>itemudalovmod</type>
            <type>relitemheadcre</type>
            <type>relitemheadmod</type>
            <type>relitemheaddel</type>
            <type>relitemdetcre</type>
            <type>relitemdetmod</type>
            <type>relitemdetdel</type>
        </injector>
</family>
<family name="itemloc">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>itemloccre</type>
            <type>itemlocmod</type>
            <type>itemlocdel</type>
            <type>itemlocreplmod</type>
        </injector>
    </family>
 <family name="merchHier"
drop-messages-of-types="divisioncre,divisionmod,divisiondel,groupcre,groupmod,grou
pdel">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>deptcre</type>
            <type>deptmod</type>
            <type>deptdel</type>
            <type>classcre</type>
            <type>classmod</type>
            <type>classdel</type>
            <type>subclasscre</type>
            <type>subclassmod</type>
```

```
            <type>subclassdel</type>
        </injector>
    </family>
<family name="order">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>pocre</type>
            <type>podel</type>
            <type>podtlcre</type>
            <type>podtldel</type>
            <type>podtlmod</type>
            <type>pohdrmod</type>
        </injector>
    </family>
<family name="prmprcchg">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>multibuypromocre</type>
            <type>multibuypromodel</type>
            <type>multibuypromomod</type>
            <type>prmcnlitemloccre</type>
        </injector>
     </family>
   <family name="clrprcchg">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>clrprcchgcre</type>
            <type>clrprcchgdel</type>
            <type>clrprcchgmod</type>
        </injector>
    </family>
<family name="regprcchg">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>regprcchgcre</type>
            <type>regprcchgdel</type>
            <type>regprcchgmod</type>
        </injector>
    </family>
   <family name="rtvreq">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>rtvreqcre</type>
            <type>rtvreqmod</type>
            <type>rtvreqdel</type>
            <type>rtvreqdtlcre</type>
            <type>rtvreqdtldel</type>
            <type>rtvreqdtlmod</type>
        </injector>
    </family>
 <family name="seeddata">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>difftypecre</type>
            <type>difftypedel</type>
            <type>difftypemod</type>
        </injector>
    </family>
 <family name="sostatus">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>sostatuscre</type>
        </injector>
</family>
 <family name="stockorder">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>socre</type>
```

```
            <type>sodtlcre</type>
            <type>sodtldel</type>
            <type>sodtlmod</type>
            <type>sohdrdel</type>
            <type>sohdrmod</type>
        </injector>
    </family>
<family name="stores”>
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>storecre</type>
            <type>storedel</type>
            <type>storemod</type>
            <type>storedtlcre</type>

            <type>storedtldel</type>

            <type>storedtlmod</type>
        </injector>
    </family>
<family name="vendor">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>vendoraddrcre</type>
            <type>vendoraddrdel</type>
            <type>vendoraddrmod</type>
            <type>vendorcre</type>
            <type>vendordel</type>
            <type>vendorhdrmod</type>
            <type>vendoroucre</type>
            <type>vendoroudel</type>
        </injector>
    </family>
<family name="wh" drop-messages-of-types="whdtlcre,whdtlmod,whdtldel">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>whcre</type>
            <type>whdel</type>
            <type>whmod</type>
        </injector>
    </family>
 <family name="rcvunitadj">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>rcvunitadjcre</type>
            <type>rcvunitadjmod</type>
        </injector>
    </family>
 <family name="dlvyslt">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>dlvysltcre</type>
            <type>dlvysltmod</type>
            <type>dlvysltdel</type>
        </injector>
    </family>
 <family name="partner">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>partnercre</type>
            <type>partnermod</type>
            <type>partnerdel</type>
            <type>partnerdtlcre</type>
            <type>partnerdtlmod</type>
            <type>partnerdtldel</type>
        </injector>
```

```
    </family>
 <family name="udas">
        <injector class="oracle.retail.sim.intrib.core.SimMessageRibInjector">
            <type>udahdrcre</type>
            <type>udahdrdel</type>
            <type>udahdrmod</type>
            <type>udavalcre</type>
            <type>udavaldel</type>
            <type>udavalmod</type>
        </injector>
    </family>
</injector_config>
```

Port Configuration

The SIM PC and handheld clients require a number of ports to be open on the SIM server in order to communicate. That means these ports will have to be opened on any firewalls between the SIM clients and the SIM server.

The following types of ports are required to be open by SIM:

- WLS HTTP port (to download the SIM client)
- WLS RMI ports (to make RMI calls from the SIM client to the SIM server)
- Wavelink server port (for the handheld devices to communicate with the Wavelink server)

The Wavelink port is defined in wavelink-startup.sh and wireless_services.cfg. See the "Wireless Server Port in wavelink-startup.sh and wireless_services.cfg" section of the "SIM Configuration Files" appendix of the *Oracle Retail Store Inventory Management Installation Guide* for more information.

The Weblogic Application Server controls the HTTP and RMI ports. The HTTP port is a single port, but the RMI ports are defined as a range of ports. These port numbers can be changed if necessary. Refer to the following documentation for descriptions and instructions on how to change the ports:

- *Weblogic Application Server Administrator's Guide*
  - Section D.2 Port Numbers (Sorted by Port Number) - Shows the port ranges assigned by default.

    http://download.oracle.com/docs/cd/B25221_04/core.1013/b25209/portnums.htm#i688124

Configuring the Transaction Timeout for SIM

This section describes how to change settings for transaction timeout.

A transaction timeout is the maximum duration, in seconds, for transactions on the application server. If the specified amount of time expires, the transaction is automatically rolled back.

The WebLogic Server EJB container automatically sets the transaction timeout if a timeout value is not defined in the deployment descriptor. The container uses the value of the Timeout Seconds configuration parameter which has default value of 30 seconds.

The default transaction timeout settings may not be sufficient for some of SIM's processes, especially batch processes.

To change the transaction timeout setting on SIM domain server, open the WebLogic Server Console, go to the JTA page for the domain SIM is installed in, and change the value in the Timeout Seconds field.

In addition to change transaction timeout at domain level, user can also modify SIM EJB deployment descriptor file to set transaction timeout on EJB level. SIM deployment descriptor (weblogic-ejb-jar.xml) has specified transaction timeout settings for following EJBs:

Example 3–15 weblogic-ejb-jar.xml

```
<weblogic-enterprise-bean>
 <ejb-name>BatchBean</ejb-name>
 <transaction-descriptor>
     <trans-timeout-seconds>3600</trans-timeout-seconds>
       </transaction-descriptor>
        </weblogic-enterprise-bean>

    <weblogic-enterprise-bean>
 <ejb-name>MpsCoordinatorBean</ejb-name>
 <transaction-descriptor>
         <trans-timeout-seconds>300</trans-timeout-seconds>
        </transaction-descriptor>
        <enable-call-by-reference>true</enable-call-by-reference>
        </weblogic-enterprise-bean>

<weblogic-enterprise-bean>
        <ejb-name>MpsWorkerBean</ejb-name>
       <transaction-descriptor>
       <trans-timeout-seconds>3600</trans-timeout-seconds>
       </transaction-descriptor>
       <enable-call-by-reference>true</enable-call-by-reference>
       </weblogic-enterprise-bean>

      <weblogic-enterprise-bean>
      <ejb-name>StockCountBean</ejb-name>
      <transaction-descriptor>
     <trans-timeout-seconds>300</trans-timeout-seconds>
      </transaction-descriptor>
      <enable-call-by-reference>true</enable-call-by-reference>
     </weblogic-enterprise-bean>

     <weblogic-enterprise-bean>
     <ejb-name>StockCountLocationBean</ejb-name>
     <transaction-descriptor>
     <trans-timeout-seconds>300</trans-timeout-seconds>
     </transaction-descriptor>
     <enable-call-by-reference>true</enable-call-by-reference>
     </weblogic-enterprise-bean>
     <weblogic-enterprise-bean>
     <ejb-name>StockCountLineItemBean</ejb-name>
     <transaction-descriptor>
     <trans-timeout-seconds>300</trans-timeout-seconds>
     </transaction-descriptor>
     <enable-call-by-reference>true</enable-call-by-reference
     </weblogic-enterprise-bean>
```

Logging Information

One of the first places to look for information concerning a problem in SIM is in the log files. Stack traces and debugging information can be found within the log files.

The log files are configured to roll over once they reach a certain size (currently 10 MB). Once a log file reaches the configured size, it will be renamed (for example, sim.log will be renamed to sim.log.1) and new log messages will be written to a new file (for example, sim.log). If there are already rolled-over logs, they will be also be renamed for example, sim.log.1 becomes sim.log.2, sim.log.2 becomes sim.log.3, and so forth). Only ten files are kept. If ten files already exist and the current file rolls over, the oldest log file is deleted.

Default Location of Log Files

The following describes the default location of the server log files and the client log files.

Server Log Files

The server log file location can be changed by changing the value of the File parameter in the sim.appender appender log4j.xml file. See Configuration Files for log4j.xml locations.

Client Log Files

Client-side log files are put in a directory called **log**, which is put wherever `user.dir` is defined in your system. For example, if you launched the Web start client with Firefox, `user.dir` is the directory where Firefox is installed. This means (depending on where you have Firefox installed) your logs could be in: `C:\Program Files\Mozilla Firefox\log\sim.log`.

To find the location of `user.dir`, double-click on the status bar at the bottom of the SIM PC client to bring up the Client Information dialog. Click the **Version** tab; one of the entries in the table is for the System Property `user.dir`. The value in the **Version** column shows the location of `user.dir` on the current client's system.

Changing Logging Levels

Sometimes it is useful to change the amount of information that the SIM server logs. There are two ways to change logging levels: editing the log4j.xml file, or using the Oracle Enterprise Manager Application Server Control user interface.

Editing log4j.xml

It is possible to change the level of any logger in the log4j.xml file. It is also possible to add new loggers if you want a certain SIM class to log more information. For more detail about loggers and logging levels, see the Log4J documentation at `http://logging.apache.org/log4j/2.x/index.html`.

Note: After changing a log level in log4j.xml the SIM server must be bounced before the change will take effect.

Activity Locking

Activity locking has been designed to be controlled from within SIM. The following example illustrates the logic of activity locking.

A user becomes involved with a warehouse delivery that includes containers with multiple items in containers; that is, a significant amount of back and forth processing between screen and server is occurring. From the GUI, a call is made to the activity lock that instructs the system that the user is working with the warehouse delivery. If some other user has the lock, the system asks the user whether he or she wishes to break it and take over. A **yes** response to the prompt implies that former owner of the lock left the lock dangling without a good reason (left to get lunch and so on). A **no** response to the prompt implies that the former owner of the lock continues to legitimately need it in place in order to finish processing.

4

Batch Processes

This chapter provides the following:

- An overview of SIM's batch processing
- A summary of SIM batch lists
- Batch scheduling and dependency list
- A description of batch detail and how to run batch processes, along with key parameters

SIM batches are executed as Java batch processes. Most of the Java batch processes engage in some processing of their own. However, the majority of work is done by services running on the SIM server; the Java batch processes make remote calls to the server to access these services.

Note the following characteristics of SIM's Java batch processes:

- They are not accessible through a graphical user interface (GUI).
- They are scheduled by the retailer.
- They are designed to process large volumes of data, depending upon the circumstances and process.

Running a Batch Process

SIM batch programs are run or scheduled through executable shell scripts (.sh files). Oracle Retail provides shell scripts for each SIM batch program.

The SIM batch program location is referred to as sim-batch-dir for the remainder of this chapter.

See the "SIM Batch Scripts" in the *Oracle Retail Store Inventory Management Installation Guide* for batch install locations.

Each batch script performs the following internally:

- Set up the class path before the Java process is run.
- Start the Java batch process.

Do the following to configure a batch environment:

1. Batch user logs in as valid batch user to the machine where SIM batch scripts are installed. The batch user must have permission to execute the SIM shell script.
2. Set JAVA_HOME environment variable and add $JAVA_HOME/bin in the PATH environment variable. For example:

```
JAVA_HOME=<jre location>
PATH=$JAVA_HOME/bin:$PATH
export PATH JAVA_HOME
```

Note: This command can be saved in a .profile file; the batch user can execute .profile before running SIM batches.

3. Execute the batch script from <*sim-batch-dir*>/bin.

For more information about batch usage, see the "Batch Details" in this chapter.

Scheduler and the Command Line

If the retailer uses a scheduler, arguments are placed into the scheduler.

If the retailer does not use a scheduler, arguments must be passed in at the command line.

Return Value Batch Standards

The following guidelines describe the function return values and the program return values that SIM's batch processes utilize:

- 0 - The function completed without error, and processing should continue normally.
- 1 - A non-fatal error occurred (such as validation of an input record failed), and the calling function should either pass this error up another level or handle the exception.

Batch Logging

Relevant progress messages are logged with regard to batch program runtime information. The location of sim batch log and logging levels can be configured in log4j.xml file which is located at <sim-batch-dir>/resources directory.

The user running the batch process must have write permission on the directory into which the sim batch log is written, or the batch process will not run. If it is not acceptable to give the batch user permission for the default log directory, log4j.xml must be configured to use a different directory.

For more information, see the "Logging Information".

Note: Some batch programs evoke Oracle stored procedure which runs on the Oracle database server, the log generated by the Oracle process may exist in different location which can be accessed by the Oracle database process. The log location is specified in batch detail section if it is different from the default batch log location.

Summary of SIM Batch List

Table 4–1 summarizes SIM's batch programs and includes a description of each batch program's business functionality. For a batch purge program list, see the "Batch Process Scheduling Notes".

Table 4–1 Batch Process Business Functionality and Dependencies

Batch Name	Description	Dependencies
AutoReceiveFinisherDeliveries	Auto-receives finisher deliveries.	No dependencies
AutoReceiveTransfers	The batch process auto-receives transfers.	No dependencies
AutoReceiveWarehouseDeliveries	The batch process auto-receives warehouse deliveries.	No dependencies
AutoReplenishCapacity	The batch process auto-replenish shop-floor according to the capacity setup.	No dependencies
AutoTicketPrint	The batch process prints tickets.	No dependencies
CleanupShelfReplenishment	The end of day batch process runs at the end of each day to reset the delivery bay and close any open pending shelf replacements.	No dependencies
ClearancePriceChange	This batch process imports the clearance price changes set up in a price management system. SIM uses this data to update the price information of the items.	No dependencies
CloseProdGroupSchedule	This batch process closes the product group schedule.	No dependencies
DeactivateOldUsers	This batch program deactivates users when their end dates have reached specified date.	No dependencies
DexnexFileParser	This batch imports the direct delivery shipment records (PO, shipment and receipt) from Dex/Nex files.	No dependencies
ExtractUnitAmountStockCount	This batch generates UnitAmount stock counts.	No dependencies
ExtractUnitStockCount	This batch generates Unit stock counts.	No dependencies
FulfillmentOrderPickReminders	This batch sends out e-mail alerts for fulfillment order picks for which create date has expired.	No dependencies
FulfillmentOrderReminders	This batch process sends out e-mail alerts for fulfillment orders for which create date has expired.	No dependencies
GenerateItemQRCodeTicket	The batch creates item tickets or shelf labels for item QR code changes.	No dependencies
ItemPriceToHistory	This batch writes the active item price records into item price history table.	No dependencies
ItemRequest	The batch process generates item requests in pending or worksheet status for item request product group schedule which was scheduled for current date	No dependencies
PosTransactionImport	This batch imports ORSPOS sale and order transactions (SIMT-LOG file) into SIM.	No dependencies
PosTransactionRetry	This batch reprocess the pos transactions which are in error state.	No dependencies
PriceChangeExtractRetry	This batch retries failed Price Change extract from previous processing which are ready for retry.	No dependencies
ProblemLineStockCount	The batch goes through the list of items in the problem line group, determining which fall within the user specified parameters (negative SOH, negative available, and so forth). The system automatically creates a stock count from those items that do fall within the parameters.	No dependencies

Table 4–1 (Cont.) Batch Process Business Functionality and Dependencies

Batch Name	Description	Dependencies
PromotionPriceChange	This batch process imports the promotional price changes setup in a price management system. SIM uses this data to update the price information of the items.	No dependencies
RegularPriceChange	This batch process imports the permanent/regular price changes setup in a price management system. SIM uses this data to update price information of the items.	No dependencies
RetailSaleAuditImport	This batch program imports sales/order transaction data (SIM-ReSA file) that originated in Oracle Retail Point-of-Service.	No dependencies
ReturnNotAfterDateAlert	This batch process warns users **x** number of days in advance that the RTV/RTW is about to reach the Not After Date and must be dispatched.	No dependencies
StockCountAuthorizeRecovery	This batch process attempts to complete a failed stock count authorization.	No dependencies
StoreSequenceImport	This batch file import sequencing information like store sequence areas and items mapped to those areas from a flat file.	No dependencies
ThirdPartyStockCountImport	This batch process imports stock count file from a third-party counting system. The stock on hand quantities are updated for the existing unit and amount stock count records in SIM.	No dependencies
TransfersOverdueBatch	This batch process sends user e-mail for dispatched transfers which have not been received after a number of days.	No dependencies
UINAttributeImport	This batch set up store UIN attributes on department/class level from the UIN Attribute import file.	No dependencies
WastageInventoryAdjustments	This batch process looks for wastage product groups that are scheduled for today and creates an inventory adjustment for each item in the scheduled product group, the wastage adjustment records are staged and published to external system (such as RMS).	No dependencies

Batch Process Scheduling Notes

Most SIM batches can be scheduled to run at any time (ad hoc) with no particular order, while some of batches might provide optimal results when batches are run in a particular order. Table 4–2 provides some scheduling recommendations:

Table 4–2 Batch Scheduling Notes

Batch Name	Schedule Type	Successor Depends on Success of Predecessor	Notes
PosTransactionImport RetailSaleAuditImport	Daily/ad hoc	Not required. The successor batch runs regardless of success/failure of the predecessor batch.	These batches should ideally be run together. Running them together to ensure inventory accuracy.
DeactivateOldUsers PurgeDeletedUsers PurgeUserPasswordHistory PurgeInvalidUserRoles PurgeUserCache	Ad hoc/Daily	Not required. The successor batch runs regardless of success/failure of the predecessor batch.	These batches should run on a continuous basis to ensure tight security and appropriate access to SIM.
CleanupShelfReplenishment	Daily		This batch should be run at least once a day if the delivery bay is used.
WastageInventoryAdjustments	Daily		This batch should run one time a day and the inventory adjustment MPS Worker must be enabled to ensure inventory accuracy.
AutoReceiveFinisherDeliveries AutoReceiveTransfers AutoReceiveWarehouseDeliveries	Daily/ad hoc	Not required. The successor batch runs regardless of success/failure of the predecessor batch.	These batches should be run at least once per day or for appropriate receipt closures.
ItemPriceToHistory	Daily		This batch should run once per day.

Batch Details

The following section summarizes SIM's batch processes and includes both an overview of each batch process business functionality, assumptions, and scheduling notes for each batch.

AutoReceiveFinisherDeliveries Batch

AutoReceiveFinisherDeliveries does the following:

- Retrieves a list of all stores.
- For each store, if the External Finisher Auto Receive store parameter is set to **Date Driven**, then the batch auto-receives all finisher deliveries that are in **New** status and whose Estimated Time of Arrival (ETA) added to the **External Finisher Auto Receive Number Of Days** is less than the current date.

Usage

The following command runs the AutoReceiveFinisherDeliveries batch:

```
AutoReceiveFinisherDeliveries.sh
```

AutoReceiveTransfers Batch

AutoReceiveTransfers does the following:

- Retrieves a list of all stores.
- For each store, if the Store Auto Receive store parameter is set to **Date Driven** and the receiving-from store is set up for auto-receiving, then the batch auto-receives all transfers that are either in Dispatched status at the sending store and In Transit status at receiving store, and whose Ship Date added to the Store Auto Receive Number Of Days store parameter value is less than or equal to the batch date.
- The batch will not auto receive the transfers in following scenarios:
  - UIN processing is enabled at Receiving Store and the transfer has UIN enabled items.
  - The transfer is locked by any other user or program.

  In these scenarios, the batch will skip the transfer and will continue with the next record.

Usage

The following command runs the AutoReceiveTransfers batch:

```
AutoReceiveTransfers.sh <ship_date>
```

Where the `ship_date` is optional and a date is not entered, then the server date is used.

AutoReceiveWarehouseDeliveries Batch

AutoReceiveWarehouseDeliveries does the following:

- Retrieves a list of all stores.
- For each store, if the Warehouse Auto Receive store parameter is set to **Date Driven**, then the batch auto-receives all warehouse deliveries that are in New status and whose Estimated Time of Arrival (ETA) added to the Warehouse Auto Receive Number Of Days store parameter value is less than or equal to the batch date.
- The batch will not auto receive the warehouse deliveries in following scenarios:
  - UIN processing is enabled at Store and the delivery has UIN enabled items.
  - The delivery is locked by any other user/program.

  In these mentioned scenarios, the batch will skip the warehouse delivery and will continue with the next record.

Usage

The following command runs the AutoReceiveWarehouseDeliveries batch:

```
AutoReceiveWarehouseDeliveries.sh <ship_date>
```

Where the `ship_date` is optional and a date is not entered, then the server date is used.

AutoReplenishCapacity Batch

The batch process looks for those product groups that are set up as shelf replenishment type that are scheduled for the current date. The batch will then create shelf replenishment records for the items defined in the product group according to their

store sequence area capacity. After creating the records, it will confirm the shelf replenishment and thus updating shop-floor quantities for the respective items.

Usage

The following command runs the AutoReplenishCapacity batch:

```
AutoReplenishCapacity.sh <date>
```

Where the date is optional and if date is not entered, then the server date is used.

AutoTicketPrint Batch

The batch process ultimately sends tickets that match the product group items setup in the store for printing. First it finds pending Item Tickets within the store that matches the Auto Ticket Print product group for the day. Then it updates the quantity of the tickets if the refresh flag was enabled for that product group.

It further sorts the tickets based on either print order (if available), area sequence, item sequence, item hierarchy or a combination of all four. After that, it consolidates the tickets; removing identified duplicates, sends the tickets for print and finally mark all the tickets within that process as print submitted.

Usage

The following command runs the AutoTicketPrint batch:

```
AutoTicketPrint.bat
AutoTicketPrint.sh
```

CleanupShelfReplenishment Batch

The end of day batch process runs at the end of each day to reset the delivery bay and close any open pending shelf replenishments. The system takes the entire inventory from the delivery bay and moves it to the back room. Any pending or in progress shelf replenishment are changed to a cancelled state. Users who are performing a shelf replenishment are kicked out of the system. That is, the batch process takes over the shelf replenishment user's application activity locking. The current user's shelf replenishment process is discarded without being saved. After the batch process is run, all shelf replenishments are either completed or cancelled, and the delivery bay has zero inventory.

Usage

The following command runs the CleanupShelfReplenishment batch job:

```
CleanupShelfReplenishment.sh
```

ClearancePriceChange Batch

This batch imports the clearance price changes from flat file into SIM item price table.

There are two phases involve in the batch process. The file load phase loads the file into price change worksheet table; the extract phase kicks off multiple threads to extract the approved worksheet records into item price table.

Usage

The following command runs the ClearancePriceChange batch:

ClearancePriceChange.sh <file_name>

Where filename (required): the file location and name of the input price change file, the file path can be absolute path or relative path to batch program.

The clearance.price.import.dir can be set up during SIM application installation, see the *Oracle Retail Store Inventory Management Installation Guide* for details.

Example:

ClearancePriceChange.sh <clearance.price.import.dir >/clearance_file1.dat.

Backup

It is customer's responsibility to backup the original data files before processing. The batch process deletes the data file after the data is loaded into the worksheet table.

File Layout

See the "Appendix: Batch File Layout Specifications"section for file layout details.

Threading

The file loading is single thread process, each file can only be loaded by a single thread, the operation is all-or-none transaction.

Once the file is loaded into worksheet table, extracting the staged worksheet records can be processed concurrently.

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see "Batch Configuration."

CloseProdGroupSchedule Batch

This batch program searches for all open product group schedules that have ended date before today (or user specified date), and change the product group schedule status to closed.

Usage

The following command runs the CloseProdGroupSchedule batch:

```
CloseProdGroupSchedule.sh <close_date>
```

Where the `close_date` is optional and a date is not entered, then the server date is used.

Error Handling and Logging

The following describes two phases:

File Loading Phase If an error occurs during the file loading phase, the batch process will terminate and no records will be loaded into worksheet table. The batch import record in BATCH_IMP_EXP table will be marked as file to stage failed. The errors will be recorded in the batch log file.

If the error is due to database is unavailable or file not found, after the database is available or the file error is corrected, then retry file load can be processed using following command.

Retry file load:

ClearancePriceChange.sh <file_name>

Extract Phase If an error occurs during the extracting worksheet phase, the batch process will mark the batch import record as import failed.

Retry extract records:

See the "PriceChangeExtractRetry Batch" for details.

DeactivateOldUsers Batch

This batch process finds active users that have passed their end date and updates their status in table SECURITY_USER to inactive.

Usage

The following command runs the DeactivateOldUsers batch:

```
DeactivateOldUsers.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

DexnexFileParser Batch

This batch imports the direct delivery shipment records (PO, shipment and receipt) from Dex/Nex files in the DEX/NEX directory into SIM.

With the uploaded data, SIM processing creates a DEX/NEX direct delivery, allowing the store user to view, edit, and confirm the information contained in the DEX/NEX file before approving it so that it can become an in progress direct delivery.

Usage

The following command runs the DexnexFileParser batch:

```
DexnexParser.sh file_name
```

Where the file name(required): is the file name which contains the data.

ExtractUnitAmountStockCount Batch

This batch program generates UnitAmount stock counts.

On a daily basis, the batch process creates the stock counts that are scheduled for the current day or future date which matches the next scheduled date. The system looks at all the scheduled stock count records and determines whether any are scheduled for today or the user-specified future date. The process creates the stock counts for each individual store. For example, if a scheduled count includes a list of five stores, then five separate stock count records are created.

If an all-location stock count is being run, the batch processing generates individual counts for every macro sequence location.

The date parameter is optional when running the Extract Stock Counts batch. If no date is provided, today's date is used.

Usage

The following command runs the ExtractUnitAmountStockCount batch:

```
ExtractUnitAmountStockCount.sh <extract_date>
```

Where the `extract_date` is optional; if specified, it must be in format of `dd/MM/yyyy`.

Primary Tables Involved

- GROUP_SCHEDULE_EXTRACT
- PRODUCT_GROUP
- PRODUCT_GROUP_HIERARCHY
- PRODUCT_GROUP_ITEM
- PRODUCT_GROUP_SCHED_STORE
- PRODUCT_GROUP_SCHEDULE
- SCHEDULE_GROUP_ITEM
- STOCK_COUNT
- STOCK_COUNT_CHILD
- STOCK_COUNT_LINE_ITEM
- STOCK_COUNT_LINE_ITEM_UIN
- STOCK_COUNT_ADHOC_CONFIG

Threading

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration".

ExtractUnitStockCount Batch

This batch program generates Unit stock counts.

On a daily basis, the batch process creates the stock counts that are scheduled for the current day or future date which matches the next scheduled date. The system looks at all the scheduled stock count records and determines whether any are scheduled for today or the user specified future date. The process creates the stock counts for each individual store.For example, if a scheduled count includes a list of five stores, then five separate stock count records are created.

If the system is configured to use unguided stock counts, the batch process does not generate multiple counts even if the item is located at multiple locations within the store.

The date parameter is optional when running the Extract Stock Counts batch. If no date is provided, today's date is used.

Usage

The following command runs the ExtractUnitStockCount batch:

```
ExtractUnitStockCount.sh <extract_date>
```

Where the `extract_date` is optional; if specified, it must be in format of `dd/MM/yyyy`.

Primary Tables Involved

- GROUP_SCHEDULE_EXTRACT

- PRODUCT_GROUP
- PRODUCT_GROUP_HIERARCHY
- PRODUCT_GROUP_ITEM
- PRODUCT_GROUP_SCHED_STORE
- PRODUCT_GROUP_SCHEDULE
- SCHEDULE_GROUP_ITEM
- STOCK_COUNT
- STOCK_COUNT_CHILD
- STOCK_COUNT_LINE_ITEM
- STOCK_COUNT_LINE_ITEM_UIN
- STOCK_COUNT_ADHOC_CONFIG

Threading

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration".

FulfillmentOrderPickReminders Batch

This batch process sends out e-mail alerts for fulfillment order picks for which create date has expired by minutes to hold customer orders before sending e-mail alert parameter value and the status is new or in progress.

Usage

The following command runs the FulfillmentOrderPickReminders batch:

```
FulfillmentOrderPickReminders.sh
```

FulfillmentOrderReminders Batch

This batch process sends out e-mail alerts for fulfillment orders for which create date has expired by minutes to hold customer orders before sending e-mail alert parameter value.

Usage

The following command runs the FulfillmentOrderReminders batch:

```
FulfillmentOrderReminders.sh
```

GenerateItemQRCodeTicket Batch

The batch looks at the item QR code table for any QR type item that has a date equal to the date for which the batch is running. The batch creates item QR code item tickets or shelf labels provided that the store options (SEND_ITEM_TICKETS_TO_TICKETING_FOR_QR_CODE_CHANGE or SEND_SHELF_EDGE_LABELS_TO_TICKETING_FOR_QR_CODE_CHANGE) are set for the store.

The batch uses the following store options:

- Send item tickets to ticketing for QR code changes.
- Send item labels to ticketing for QR code changes.

Usage

The following command runs the batch:

GenerateItemQRCodeTicket.sh <batch_date>

Where the batch_date is optional and a date is not entered, then the server date is used.

ItemPriceToHistory Batch

This batch writes the active item price records into item price history table. After the active item prices are recorded in the item price history table, the batch updates the ITEM_PRICE table statuses as completed for these records.

Usage

The following command runs the batch:

ItemPriceToHistory.sh <date in format of dd/MM/yyyy >

Where the date (optional) is the date which the price effectives at or pricing ends at.

If date is not provided, then the batch client current time is used.

ItemRequest Batch

The batch process looks for those product groups that are set up as item request type that are scheduled for the current date. It generates the item request (with items and quantities) in a pending or worksheet status. The user (for example, a manager) can then add items, delete items, change quantities, and so on before submitting the data to the merchandising system. The merchandising system can generate POs or warehouse to store transfers as applicable. The batch also cancels out the expired item requests.

Usage

The following command runs the ItemRequest batch:

```
ItemRequest.sh
```

Threading

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration".

PosTransactionImport Batch

ORPOS can post transactions to ORSIM either through web service route or through a flat file.

PosTransactionImport batch imports pos transaction records from the flat file (SIMT-LOG file) that came from ORPOS into the SIM database staging table where polling timer framework will pick those staged request and update the stock tables in ORSIM.

Usage

From SIM batch location, run command.

posTransactionImport.sh <filename1>

Where filename(required): is the file name input simt_log file that contains the transaction data. The path of the file can be absolute or relative to the batch client.

The <pos.sale.import.dir> can be set up during SIM application installation, see the *Oracle Retail Store Inventory Management Installation Guide* for details.

Example: PosTransactionImport.sh <pos.sale.import.dir>/simtlog_file1.dat.

Batch Detail

posTransactionImport batch process takes the sales/order transaction data and stage them to the ORSIM database staging table from where they are picked up by the polling timer framework to update the store item's inventory buckets (for example, store item's total quantity, shop floor quantity), if applicable.

The file will contain both sale and order transactions. The batch will assign separate request IDs to sales and order transactions.

For sale transactions, a single request ID cannot contain more than MAX_VALUE = 500 transaction line items with an exception that a single transaction ID cannot span across multiple request IDs.

For order transactions, a single request ID cannot contain more than MAX_VALUE = 500 transaction line items with an exception that a single customer order ID cannot span across multiple request IDs.

The file contains transactions for a single store.

Threading

posTransactionImport batch supports one file per thread. The operation is all or none transaction. For multiple file execution you can run multiple instances of the batch file providing each process with a separate audit file.

Error Handling, Logging, and File Archiving

posTransactionImport batch process will notify the user with an error message if the file staging fails. The staging process is all or none transaction so if an error occurs during the batch process, none of the transactions in the file will be staged. The user will need to rerun the same file again after resolving any errors.

PosTransactionRetry Batch

This batch processes previous failed POS transaction records (POS_TRANSACTION) which are ready to be retried (status = 3).

With the POSTransactionService Web service/SIMT_LOG file integration between Oracle Retail Point-of-Service and SIM, the POS Transactions (sale, return, void of sale, void of return order new, order cancel, order fulfill) are integrated into SIM and inventory updates take place in SIM.

If there is an error (status = 2) because of incorrect data, the PosTransactionRetry batch can be run to process the those failed records providing that user has corrected the errors and changed the failed transaction line items status to 3 (RETRY).

Usage

The following command runs the PosTransactionRetry batch:

PosTransactionRetry.sh <store_id>

Where store_id (required): is the store ID for which you want to process the error POS transactions which are read to be retried.

PriceChangeExtractRetry Batch

This batch re-processes (extracts) the approved price change worksheet records which were missed from the previous price change batch executions (such as database server was down or dependent data was missing) after the issues are resolved.

Run this batch if price change batch (ClearancePriceChange, PromotionPriceChange, RegularPriceChange) encounters an error.

For batch import in question, user will need to inspect the price change worksheet records for the batch import). For records which are in status of failed extract (status = 7) or rejected (status = 4) and are recoverable once the errors are resolved (such as dependent data is now available, or db server is up), user needs to set the status back to 2 (approved) for re-processing.

Usage

The following command runs the batch:

PriceChangeExtractRetry.sh <extract_id>

Where extract_id (required) is the extract ID in PRICE_CHANGE_WORKSHEET table.

ProblemLineStockCount Batch

Before the batch process runs, the retailer establishes a group of items and item hierarchies (by associating them to the problem line group type) and selects applicable parameters (negative SOH, negative available, and so on). The problem line batch process goes through the list of items in the group, determining which fall within the parameters. The system automatically creates a stock count from those items that do fall within the parameters.

If an item is a problem line item (negative inventory for example) on a stock count, and the user does not get the chance to perform the stock count on it that day, the next day the item may no longer be a problem line (positive inventory). However, the system continues to create a stock count for that item because a problem existed at one time.

Usage

The following command runs ProblemLineStockCount batch:

```
ProblemLineStockCount.sh
```

Threading

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration."

PromotionPriceChange Batch

This batch imports the promotion price changes from flat file into SIM item price table.

There are two phases involve in the batch process. The file load phase loads the file into price change worksheet table; the extract phase kicks off multiple threads to extract the approved worksheet records into item price table.

Usage

The following command runs the PromotionPriceChange batch:

```
PromotionPriceChange.sh <file_name>
```

Where filename (required): import file directory and name of the import file.

The < promotion.price.import.dir> can be set up during SIM application installation, see the *Oracle Retail Store Inventory Management Installation Guide* for details.

Example:

PromotionPriceChange.sh <promotion.price.import.dir> /promo_file1.dat.

Backup

It is customer's responsibility to backup the original data files before processing. The batch process deletes the data file after the data is loaded into the worksheet table.

File Layout

See the "Appendix: Batch File Layout Specifications" section for file layout details.

Threading

The file loading is single thread process, each file can only be loaded by a single thread, the operation is all-or-none transaction.

Once the file is loaded into worksheet table, extracting the staged worksheet records can be processed concurrently.

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration".

Error Handling and Logging

The following describes the phases:

1. File Loading Phase

   If an error occurs during the file loading phase, the batch process will terminate and no records will be loaded into worksheet table. The batch import record in BATCH_IMP_EXP table will be marked as file to stage failed. The errors will be recorded in the batch log file.

   If the error is due to database is unavailable or file not found, after the database is available or the file error is corrected, then retry file load can be processed using following command:

   Retry file load:

   PromotionPriceChange.sh <file_name>

2. Extract Phase

   If an error occurs during the extracting worksheet phase, the batch process will mark the batch import record as import failed.

   Retry extract records:

   See the "PriceChangeExtractRetry Batch" batch for details.

RegularPriceChange Batch

This batch imports the regular price changes from flat file into SIM item price table.

There are two phases involve in the batch process. The file load phase loads the file into price change worksheet table; the extract phase kicks off multiple threads to extract the approved worksheet records into item price table.

Usage

The following command runs the RegularPriceChange batch:

RegularPriceChange.sh <file_name>

Where filename (required): import file directory and name of the import file.

The <regular.price.import.dir> can be set up during SIM application installation, see the *Oracle Retail Store Inventory Management Installation Guide* for details.

Example:

RegularPriceChange.sh <regular.price.import.dir>/regular_file1.dat.

Backup

It is customer's responsibility to backup the original data files before processing. The batch process deletes the data file after the data is loaded into the worksheet table.

File Layout

See the "Appendix: Batch File Layout Specifications" section for file layout details.

Threading

The file loading is single thread process, each file can only be loaded by a single thread, the operation is all-or-none transaction.

Once the file is loaded into worksheet table, extracting the staged worksheet records can be processed concurrently.

The number of concurrent extracting processes is configured through the batch.cfg file. For details, see the "Batch Configuration".

Error Handling and Logging

The following describes the phases:

1. File Loading Phase

   If an error occurs during the file loading phase, the batch process will terminate and no records will be loaded into worksheet table. The batch import record in BATCH_IMP_EXP table will be marked as file to stage failed. The errors will be recorded in the batch log file.

   If the error is due to database is unavailable or file not found, after the database is available or the file error is corrected, then retry file load can be processed using following command:

   Retry file load:

   RegularPriceChange.sh <file_name>

2. Extract Phase

If an error occurs during the extracting worksheet phase, the batch process will mark the batch import record as import failed.

Retry extract records:

For details, see the "PriceChangeExtractRetry Batch".

RetailSaleAuditImport Batch

This batch program imports sales/order transaction data (SIM-ReSA File) that originated in Oracle Retail Point-of-Service. The external audit system will provide in its sales upload file a percentage or quantity that indicates how much the inventory needs to be reduced by, in addition to the sold quantity.

For example, meat will become lighter as fluids evaporate. Other items, for example cheese or ham, will only be reduced when of the outside layers are cut off to sell the item.

SIM takes the sales transaction data to update the store item's inventory buckets. From the batch program, SIM learns about inventory movement (that is, what is sold, what is returned, what is reserved and what is fulfilled). Once SIM attains the data, SIM assumes that sales should be taken from the store's shelf-related inventory buckets. This assumption is important to SIM's shelf replenishment processing. Similarly, SIM assumes that returns should go to the backroom bucket; the system's logic is that returns must be inspected.

The batch also writes each failure record into a transaction log table.

Usage

From SIM batch location, run command:

RetailSaleAuditImport.sh <filename1>

Where filename(required): is the file name input ReSA file that contains the audit data.

The file path can be either absolute path or relative path to batch program.

Example:

RetailSaleAuditImport.sh <BATCH_DIR>/input/ReSA/audit_file1.dat

Batch Detail

RetailSaleAuditImport takes the sales/order transaction data and stage them to the ORSIM database staging table from where they are picked up by the polling timer framework to update the store item's inventory buckets (for example, store item's total quantity, shop floor quantity), if applicable.

The file will contain both sales and order transactions. Request IDs are assigned to the transactions in such a way that a single request ID will not contain more than MAX_ SIZE=500 records with an exception that a single transaction ID should not span across multiple request IDs.

Threading

RetailSaleAuditImport batch supports one file per thread. The operation is all or none transaction. For multiple, file execution you can run multiple instances of the batch file providing each process with a separate audit file.

Error Handling, Logging and File Archiving

RetailSaleAuditImport will notify the user with an error message if the file staging fails. The staging process is all or none transaction so if an error occurs during the batch process, none of the transactions in the file will be staged. The user will need to rerun the same file again after resolving any errors.

ReturnNotAfterDateAlert Batch

This batch process warns users a number of days in advance that the RTV/RTW is about to reach the **Not After** date and must be dispatched. The value for the number of days of advance warning is configurable using the system's administration screens.

Usage

The following command runs the ReturnNotAfterDateAlert batch:

```
ReturnNotAfterDateAlert.sh
```

StockCountAuthorizeRecovery Batch

This batch process looks for stock counts that are stuck in Authorize Processing state. This is a unique state that appears when an error occurs during the final processing of a stock count. The batch attempts to fully authorize the stock count. Errors that occur during the batch process are logged to the server error logs and will indicate the reason for any further processing failures. Successfully authorized stock counts will move to authorized completed state.

Usage

The following command runs the StockCountAuthorizeRecovery batch:

```
StockCountAuthorizeRecovery.sh
```

StoreSequenceImport Batch

This batch imports store sequencing information from a flat file. Before importing, the batch will delete the existing sequencing information including sequence items and sequence areas excluding no-location store area which is the default store sequence area.

Usage

The following command runs the StoreSequenceImport batch:

```
StoreSequenceImport.sh <filename1>
```

Where filename(required): is the file name input that contains the sequence data.

The file path can be either absolute path or relative path to batch program.

Example:

RetailSaleAuditImport.sh <BATCH_DIR>/input/Sequencing/store-sequence.dat

ThirdPartyStockCountImport Batch

This batch imports the stock count quantities which is setup in SIM and physical counting is conducted by a third party. The batch updates the store stock on hand quantities; invalid records are saved in the rejected item table.

For Auto-authorize Unit Amount stock count, the batch process also invokes the stock count authorization process automatically, the stock count authorization process will generate stock count result export file which can be imported by other merchandise system (such as RMS), if applicable.

For non-auto authorize stock count, after import batch complete, user will needs to go to SIM stock count authorization screen to manually authorize the stock count, the stock count authorization process will generate stock count result export file which can be imported by other merchandise system (such as RMS), if applicable.

Usage

ThirdPartyStockCountImport.sh <file_name>

Where filename (required): The import file directory and name of the import file.

The <stock.count.import.dir> can be set up during SIM application installation, see the *Oracle Retail Store Inventory Management Installation Guide* for details.

Example:

ThirdPartyStockCountImport.sh < stock.count.import.dir>/stock_count_imp1.dat

---

Note:
Export File Location

For auto-authorized unit amount stock count type, the batch will automatically invokes the stock count authorization process which generates the stock count results export file.

The export file location can be set up during SIM database installation. See the *Oracle Retail Store Inventory Management Installation Guide* for details.

If the export file location was not set up at the time as the SIM database was installed, then the following steps need to be set up by a system administrator and DBA.

The file location can be found by running the following query:

select * from dba_directories where directory_name = 'STOCK_COUNT_UPLOAD_DIR';

Export File Location Setup

1. System administrator

   Setup physical file location on a server location where SIM database process can have access to, grant operation system permission read/write to SIM application schema user.

2. Database administrator

   a. Create directory object in SIM database:

      Create or replace directory STOCK_COUNT_UPLOAD_DIR as '&batch_stockcount_upload_dir';

   b. Grant privileges:

      Grant read, write on directory STOCK_COUNT_UPLOAD_DIR to &&sim_user_uppercase;

Primary Tables Involved

These following are the primary tables involved:

- STOCK_COUNT_IMPORT
- STOCK_COUNT_REJECTED_ITEM
- STOCK_COUNT
- STOCK_COUNT_CHILD
- STOCK_COUNT_LINE_ITEM_UIN

Integration Assumptions

- RMS provides an item export file to external stock counting prior to the count in order for external stock counting to validate the items that are scanned.
- The items coming from external stock counting are identified based on an RMS item number (for example, an RIN, UPC, or other number set up in RMS).
- All quantities from external stock counting are assumed to be in the item's standard unit of measure (UOM) as established by RMS (for example, units, KG, and so on).
- The external stock counting file sends the total quantity counted for each item, regardless of whether the item was counted in several areas of the store (rolled up total by item).
- For items that exist in the SIM stock count records but do not have a counted quantity sent back from the external stock counting system, SIM assumes a count quantity of 0, and set this value on the stock count record.
- For items that have a SOH quantity in SIM but have a stock counting count of 0, the discrepancy check uses the variance units (not the variance percentage) value to determine whether the item is discrepant, user can view the discrepant items through Stock Count Rejected Items PC screen after batch completes.

TransfersOverdue Batch

This batch process sends user e-mail for dispatched transfers which have not been received after a number of days. The value for the number of days of e-mail alert is configurable using the system's administration screen.

Usage

The following command runs the TransfersOverdue batch:

```
TransfersOverdueBatch.sh
```

UINAttributesImport Batch

This batch is used for process UIN Attribute file (see the Appendix UIN Attribute File Specification) to set up store item UIN admin records on department/class level for stores which system configurations have UIN_PROCESSING_ENABLED is true and AUTO_DEFAULT_UIN_ATTRIBUTES value is true.

The batch inserts/updates the STORE_UIN_ADMIN_DEPT table for the specified department/class, and insert into STORE_UIN_ADMIN_ITEM table for each item within the merchandise hierarchy, it then marks the store item as UIN required in STORE_ITEM table.

This batch can be used to set up store item admin records after SIM initial data seeding from RMS (Retail Merchandise System); it can also be used to set up bulk store UIN admin records by department/item level as an alternate way to the SIM GUI for setting up UIN attributes.

Usage

The following command runs the UpdateUINAttributes batch:

```
UINAttributeImport.sh <file_name>
```

Where filename (required): the file location and name of the import stock count file. The file path can be absolute path or relative path to batch program.

WastageInventoryAdjustments Batch

This batch process looks for wastage product groups that are scheduled for today and creates an inventory adjustment for each item in the product group. The batch process uses amounts based on percentage/units. Note that if both a percentage and unit exist, the batch process applies the least amount of the two. For example, consider an item with a stock on hand value of 100. If the two values are 10% and 5 units, the batch process would create an inventory adjustment of 5 units for the item.

The batch process creates a completed inventory adjustment record using the adjustment reason of Shrinkage (code = 1) for each item that is published to the merchandising system.

Note: Wastage is not run for items that require UINs.

Usage

Following command runs the WastageInventoryAdjustments batch:

```
WastageInventoryAdjustments.sh
```

After the batch process is complete, the retailer must run another batch WastageInventoryAdjustmentPublishJob.sh to publish the inventory adjustment generated by the above batch to the merchandising system.

WastageInventoryAdjustmentPublishJob Batch

The batch process picks up all items that were flagged for publishing to the merchandising system. After an item is published, the flag is reset.

Usage

Following command runs the WastageInventoryAdjustmentPublishJob batch:

```
WastageInventoryAdjustmentPublishJob.sh
```

5

Data Purge

Data purging is based upon a data-retention schedule whereupon all data existing prior to the computed date are purged. The data within this timeframe must meet data integrity constraints as well as functional requirements.

SIM Data purging programs are meant to provide the basic implementation of logical selection of sets of data based on functional requirement and data retention periods. The data retention periods are defined in SIM system configuration table. Logical sets of data may exist in multiple tables.

With this assumption, SIM purge shell scripts execute the java EJB which calls the oracle PL/SQL stored procedures to delete the records. The purging logics are resides in PL/SQL stored procedures, Customer may choose the their preferred purging strategy and implement their own purging implementation.

Data Purge Logging

Relevant progress messages are logged with regard to purge batch program runtime information. The location of sim batch log and logging levels can be configured in log4j.xml file which is located in <sim_batch_location>/resources/conf/log4j.xml.

The user running the batch process must have write permission on the directory into which the sim batch log is written, or the batch process will not run. If it is not acceptable to give the batch user permission for the default log directory, log4j.xml must be configured to use a different directory.

Note: The purge PL/SQL programs may save the detailed exceptions into PURGE_ERROR_LOG table. DBA needs to purge the data periodically.

Summary of SIM Purge List

The following section summarizes SIM's data Purge programs and includes both an overview of each purge program business functionality, and scheduling notes.

Table 5–1 Data Purge Program List and Dependencies

Purge Name	Description	Dependencies
PurgeAdHocStockCount	Deletes AD HOC stock counts with a status of **in progress** which schedule date was X days in the past.	No dependencies
PurgeAll	Deletes records from the SIM application that meet certain business criteria.	No dependencies
PurgeAudits	This batch process deletes audits.	No dependencies
PurgeBatchImpExp	This batch process deletes batch import export records.	No dependencies
PurgeCompletedUINDetail	This batch deletes completed UIN Detail records.	No dependencies
PurgeDeletedUsers	This batch program deletes users that have a status of deleted and have been held in the system for a number of system defined days.	No dependencies
PurgeDSDReceivings	This batch process deletes the Direct Store Delivery receiving.	No dependencies
PurgeFulfillmentOrders	This batch deletes closed fulfillment order records.	No dependencies
PurgeInvalidUserRoles	This batch program removes all expired or orphaned roles from the users in the system.	No dependencies
PurgeInventoryAdjustments	This batch process deletes inventory adjustments.	No dependencies
PurgeInventoryAdjustTemplate	This batch deletes inventory adjustment templates.	No dependencies
PurgeItem	This batch program deletes the items that are in deleted status (D).	No dependencies
PurgeItemBaskets	This batch process deletes item baskets.	No dependencies
PurgeItemHierarchy	This batch deletes unused and deleted item hierarchies.	No dependencies
PurgeItemPrice	This batch deletes expired or deleted item prices.	No dependencies
PurgeItemRequests	This batch process deletes item requests.	No dependencies
PurgeItemTickets	This batch process deletes item tickets.	No dependencies
PurgeLockings	This batch process deletes lockings.	No dependencies
PurgePriceChangesWorksheet	This batch deletes completed/rejected price change work sheet records.	No dependencies
PurgePriceHistories	This batch process deletes price histories.	No dependencies
PurgePurchaseOrders	This batch process deletes purchase order records.	No dependencies
PurgeReceivedTransfers	This batch process deletes received transfers.	No dependencies
PurgeRelatedItems	This batch will delete related items.	No dependencies
PurgeResolvedUINProblems	This batch process deletes resolved item serial number problems.	No dependencies
PurgeSalesPosting	This purges the sales, returns, void sales and void returns transaction from the staging table.	No dependencies
PurgeShelfReplenishment	This batch deletes completed or canceled Shelf replenishment records.	No dependencies
PurgeStagedMessage	This batch process removes processed integration staging records (MPS Staged Messages).	No dependencies
PurgeStockCounts	This batch process deletes stock counts.	No dependencies
PurgeStockReturns	This batch process deletes stock returns.	No dependencies

Table 5–1 (Cont.) Data Purge Program List and Dependencies

Purge Name	Description	Dependencies
PurgeStockItemStockHistory	This batch will delete store item stock history inventory movement records.	No dependencies
PurgeTemporaryUINDetail	This batch process deletes temporary UIN detail records.	No dependencies
PurgeUINDetailHistories	This batch process deletes UIN detail history records (UIN Audit information).	No dependencies
PurgeUserCache	This batch program deletes expired user cache records.	No dependencies
PurgeUserPasswordHistory	This batch program deletes user password history records.	No dependencies
PurgeWHDReceivings	This batch process deletes the Warehouse delivery receiving records.	No dependencies

Purge Scheduling Notes

Most SIM purge programs can be scheduled to run at any time (ad hoc) with no particular order, while some of purge programs may need to run in a particular order to provide optimal results. Table 5–2, " Purge Scheduling Notes" provides some scheduling recommendations:

Table 5–2 Purge Scheduling Notes

Batch Name	Schedule Type	Successor Depends on Success of Predecessor	Notes
DeactivateOldUsers PurgeDeletedUsers PurgeUserPasswordHistory PurgeInvalidUserRoles PurgeUserCache	Ad hoc	Not required. The successor batch runs regardless of success/failure of the predecessor batch.	These batches should run on a continuous basis to ensure tight security and appropriate access to SIM.
PurgeDSDReceivings PurgePurchaseOrders PurgeReceivedTransfers PurgeStockReturns PurgeWHDReceivings PurgeFulfillmentOrders	Ad hoc	Not Required	Run these purge batches in logical order to provide optimal results.

Table 5–2 (Cont.) Purge Scheduling Notes

Batch Name	Schedule Type	Successor Depends on Success of Predecessor	Notes
PurgePriceChangesWorksheet PurgeItemPrice	Daily	Not Required	These batch should run at least once per day to remove the expired item price from the transaction table to ensure to optimized price retrieving response time.
PurgeItem	Ad hoc	Not Required	This purge should run after all other purges.
PurgeItemTicket	Ad hoc	Not Required	It is recommended to purge the printed tickets regularly to optimize the overall daily ticket printing process.

Purge Details

The following section summarizes SIM's purge processes and includes both an overview of each purge process business functionality, assumptions, and scheduling notes for each purge program.

PurgeAdHocStockCount Batch

This batch program deletes ad hoc stock counts with a status of **in progress**. Any ad hoc stock count with a creation date/time stamp older than the **Days to Hold In Progress Ad Hoc Counts** parameter value will be deleted. For example, the default value is **1**. If the batch program is run with the default value, the batch program would delete all in-progress counts more than 24 hours old.

Usage

```
PurgeAdHocStockCount.sh
```

PurgeAll Batch

This process deletes records from the SIM 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).

It is the wrapper batch which contains all purge batches which listed in the purge list.

Usage

```
PurgeAll.sh <purge_date>
```

Where purge_date is optional, date format must be dd/MM/yyyy if purge_date is specified.

PurgeAudits Batch

This batch process deletes audit records. Any audit record with a create date/timestamp older than the Days To Hold Audit Records parameter value is deleted. For example, if the default value is 30 and the batch program is run with the default value, the batch program would delete all the audit records that are more than 30 days old.

Usage

```
PurgeAudits.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeBatchImpExp Batch

This batch process deletes batch import export records. Any import export record with an end date/timestamp older than the Days To Hold Completed Staging Records parameter value and with the Status value of 2 (COMPLETED) is deleted. For example, if the default value is 30 and the batch program is run with the default value, the batch program would delete all the records that are more than 30 days old and are in completed status.

Usage

PurgeBatchImpExp.sh <purge_date>

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeCompletedUINDetail Batch

This batch program deletes completed UIN Detail records. A completed UIN is any UIN with a status of Removed from Inventory, Missing, Sold, Shipped to Vendor, or Shipped to Warehouse. Any UIN detail record with a complete status and update date at least *X* days in the past (where *X* is with system parameter DAYS_TO_HOLD_COMPLETED_UINS) will be deleted from ITEM_UIN and ITEM_UIN_PROBLEM table.

Usage

```
PurgeCompletedUINDetail.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy format if purge_date is specified.

PurgeDeletedUsers Batch

This batch process finds users marked as deleted with an end date that is at least *X* days in the past (where *X* is the system parameter SECURITY_DAYS_TO_HOLD_DELETED_USERS). These users and all associated data are deleted from SECURITY_USER_ROLE, SECURITY_USER_STORE, SECURITY_USER_PASSWORD, and SECURITY_USER tables.

Usage

```
PurgeDeletedUsers.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeDSDreceivings Batch

This batch process deletes the Direct Store Delivery receivings.

Any DSD record which is in Closed/Cancelled status and which has a complete date older than Days to Hold Received Shipments is an eligible record for purge.

However, before a DSD record is purged, checks are made to ensure that the purchase order associated with a particular DSD is also completed and is older than Days to Hold Purchase Orders.

In effect a DSD record can be purged only if its associated PO records can be purged.

Usage

```
PurgeDSDReceivings.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeFulfillmentOrders Batch

This batch process will delete all the fulfillment order records which are not in New or In Progress status and for which the update date has expired the purge_date by number of days more than Days to Hold Customer Order parameter value. Additionally, only those fulfillment orders will be deleted for which customer order ID and fulfillment order ID combination does not exist for any Transfer, Return, Purchase Order, and Warehouse delivery transaction.

Usage

```
PurgeFulfillmentOrders.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeInvalidUserRoles Batch

This batch program removes all expired user roles and orphaned user roles (roles that were deleted by removing a store) from the SIM system.

The batch process finds user role assignments that have an end date that is at least *X* days in the past (where *X* is specified by the system parameter SECURITY_DAYS_TO_HOLD_EXPIRED_USER_ROLES), and deletes these expired role assignments. The users (excluding super users) with role assignments that have no matching store assignments (orphaned role assignments) are also deleted from SECURITY_USER_ROLE table.

Usage

```
PurgeInvalidUserRoles.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeInventoryAdjustments Batch

This batch process deletes inventory adjustments. Any inventory adjustment record with a create date/timestamp older than Days To Hold Completed Inventory Adjustments parameter value will be deleted. For example, the default value is 30. If the batch program is run with the default value, the batch program would delete all the inventory adjustment records, which are more than 30 days old.

Usage

`PurgeInventoryAdjustments.sh <purge_date>`

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeInventoryAdjustTemplate Batch

This batch process deletes inventory adjustment template records for which the status is cancelled and the record (approve date) has satisfied the DAYS_TO_HOLD_CANCELLED_TEMPLATE value which is specified in CONFIG_SYSTEM table.

Usage

PurgeInventoryAdjustTemplate.sh <purge_date>

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeItem Batch

This batch program deletes items with a status of Delete (D).

There are two segments in the PurgeItem Batch which do the following different tasks:

- Validate if the Item should be deleted.
- Delete item from all associated tables if validation check is passed.
- Validate if the item should be deleted. The Validations include:
  - If SOH of item, item parent and item grandparent is 0.
  - If any transfers exist for item, item parent and item grandparent.
  - If any RTV exists for item, item parent and item grandparent.
  - If any Inventory adjustment exists for item, item parent and item grandparent and so on.
  - If any Item Basket exists for the item.
  - If any Product Group exists for the item.
  - If any Stock Count exists for the item.
  - If any Store Order exists for the item.
  - If any Item Request exists for the item.
  - If any Direct Store Delivery exists for the item.
  - If any Warehouse Delivery exists for the item.
- Delete item from all associated table. If the validations checks are met, the records related to the item which is marked for the purge action are deleted.

Usage

```
PurgeItem.sh
```

PurgeItemBaskets Batch

This batch process deletes item baskets. Any item basket record with a process date or timestamp older than batch date value is deleted.

Usage

```
PurgeItemBaskets.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeItemPrice Batch

This batch process purges records which were expired or were marked as deleted from ITEM_PRICE table based on the retention period. The retention period is specified by system configuration parameter DAYS_TO_HOLD_EXPIRED_ITEM_PRICE.

Rules defining records to be purged:

Regular Price Change: Has status of completed or deleted, effective date was X number of days in the past (relative to the specified date if specified). At any given time, at least one completed latest regular price must be retained in ITEM_PRICE for a store item.

Promotion Change: Has status of completed or deleted, and end date is number of days in the past (relative to the specified date if specified).

Clearance Change: Has status of completed or deleted, and end date is number of days in the past (relative to the specified date if specified).

Usage

```
PurgeItemPrice.sh <date>
```

Where date is optional and the date format must be dd/MM/yyyy if date is specified.

If date is not provided, then the current time on the batch client is used.

PurgeItemHierarchy Batch

This batch process purges all Item Hierarchies that is in delete status. It also records from referencing tables such as STOCK_COUNT_ADHOC_CONFIG and ITEM_HIERARCHY_ATTRIB. The batch process logs errors with executing the batch otherwise it returns a success code.

Usage

```
PurgeItemHierarchy.sh
```

PurgeItemRequests Batch

This batch process deletes item requests which are in Cancelled/Completed status. Any item request record with a process date/timestamp older than DAYS_TO_HOLD_CANCELLED_ITEM_REQUESTS system configuration value will be deleted. For example, the default value is 30. If the batch program is run with the default value, the

batch program would delete all the item request records, which are more than 30 days old.

Usage

```
PurgeItemRequests.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeItemTickets Batch

This batch process deletes item tickets which were printed or canceled at least X days in the past (where X is the system parameter DAYS_TO_HOLD_ITEM_TICKETS). It is recommended to purge the printed tickets regularly to optimize the overall daily ticket printing process.

Usage

```
PurgeItemTickets.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

Primary Tables Involved

These following are the primary tables involved:

- ITEM_TICKET
- ITEM_TICKET_UIN
- ITEM_TICKET_UDA
- TICKET_PRINT
- TICKET_PRINT_LINE_ITEM
- TICKET_REQUEST

PurgeLockings Batch

This batch process deletes lockings records from ACTIVITY_LOCK table. Any lock record with a lock date/timestamp older than DAYS_TO_HOLD_SHELF_REPLENISHMENT system configuration value will be deleted. For example, the default value is 30. If the batch program is run with the default value, the batch program would delete all the shelf replenishment records, which are more than 30 days old.

Usage

```
PurgeLockings.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgePriceChangeWorksheet Batch

This batch process deletes price change worksheet records which are in Rejected/Completed status. Any price change record with an effective date/timestamp older than Days To Hold Price Changes parameter value will be

deleted. For example, the default value is 30. If the batch program is run with the default value, the batch program would delete all the price change records, which are more than 30 days old.

Usage

```
PurgePriceChangeWorksheet.sh <purge_date>
```

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgePriceHistories Batch

This batch process deletes price histories. At least a minimum of 4 historical prices are maintained for an item/store. Days To Hold Price History will determine the number of days that price histories can be kept in the database.

Usage

```
PurgePriceHistories.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgePurchaseOrders Batch

This batch process deletes closed purchase order records. The purchase order records which are in closed status and complete date is at least *X* days in the past (where *X* is system parameter DAYS_TO_HOLD_COMPLETED_PURCHASE_ORDERS) are deleted from the database.

Usage

```
PurgePurchaseOrders.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeReceivedTransfers Batch

This batch process deletes received transfers. The transfer in and transfer out transactions will be purged from the database. The transfer out transactions which are in Rejected/Cancelled Request/Received/Cancelled Transfer will be purged if the records are older than Days To Hold Received Transfer Records parameter. Also, the Purge Received Transfers parameter must be set to **Yes** in the admin screen to enable purging of the received transfers.

Usage

```
PurgeReceivedTransfers.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeRelatedItems Batch

This batch process deletes the related items for which the end date has expired for more than Days To Hold Related Items system configuration value.

Usage

PurgeRelatedItems.sh <purge_date>

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeResolvedUINProblems Batch

This batch process deletes resolved UIN exception records. UIN exception records with status of resolved and resolved date is at least *X* days in the past (where *X* is system parameter DAYS_TO_HOLD_RESOLVED_UIN_EXCEPTIONS) are deleted from ITEM_UIN_PROBLEM table.

Usage

```
purgeResolvedUINProblems.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeSalesPosting Batch

This batch process deletes the Point-of-Service transaction from the Oracle Retail Point-of-Service transaction staging table. It reads the Days to Hold Sales Posting configuration parameter and all the transactions which are present beyond the configuration parameter are deleted. It also purges the POS transaction logs for the request IDs that are in processed status.

Usage

```
PurgeSalesPosting.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeShelfReplenishment Batch

This batch process deletes shelf replenishment lists which are in Completed/Cancelled state. Any shelf replenishment list record with a status date/timestamp older than Days To Hold Shelf Replenishment parameter value will be deleted. For example, the default value is 1. If the batch program is run with the default value, the batch program would delete all the pick list records, which are more than a day old.

In addition, the batch will also delete shelf adjustment lists which are in Complete state. Any shelf adjustment record with a update date/timestamp older than Days To Hold Shelf Adjustment Lists parameter value will be deleted.

Usage

```
PurgeShelfReplenishment.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeStagedMessage Batch

This batch finds integration staging records that are marked as processed or deleted, and update date is at least *X* days in the past (where *X* is the system parameter DAYS_

TO_HOLD_COMPLETED_STAGING_RECORDS), the batch process deletes these records from MPS_STAGED_MESSAGE table.

Rebuilding the indexes on the MPS_STAGED_MESSAGE table each day is recommended after batch process completes.

Usage

```
PurgeStagedMessageJob.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

Note: The optional date is the point of reference the script should use. The script uses current date for the point of reference by default, but can be run from any reference point.

Therefore, if purging records that are 10 days old, and running the script without the optional date, the process removes all records older than 10 days from the current date. If the optional date argument is specified, records 10 days older than the specified optional date are purged.

PurgeStockCounts Batch

This batch process deletes stock counts which are in Completed/Cancelled status. Any stock count with a schedule date/timestamp older than Days To Hold Completed Stock Counts parameter value will get deleted. For example, the default value is 30.If the batch program is run with the default value, the batch program would delete all the stock return records, which are more than 30 days old.

Usage

```
PurgeStockCounts.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeStockReturns Batch

This batch process deletes stock returns which are in Dispatched/Cancelled status. Any stock return record with a completed date/timestamp older than DAYS_TO_HOLD_RETURNS system configuration value will be deleted. For example, the default value is 30.If the batch program is run with the default value, the batch program would delete all the stock return records, which are more than 30 days old

Usage

```
PurgeStockReturns.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeStoreItemStockHistory Batch

This batch process deletes store item stock history records which have been kept for more than Days To Hold Transaction History which is specified by system configuration. For example, the default value is 30. If the batch program is run with

the default value, the batch program would delete all the stock history records, which are more than 30 days old.

Usage

PurgeStoreItemStockHistory.sh <purge_date>

Where purge_date is optional and the date format must be dd/MM/yyyy if purge_date is specified.

PurgeTemporaryUINDetail Batch

This batch process deletes temporary UIN detail records. UIN detail records with no status and update date is at least *X* days in the past (where *X* is system parameter DAYS_TO_HOLD_TEMPORARY_UINS) are deleted from ITEM_UIN table.

Usage

```
PurgeTemporaryUINDetail.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeUINDetailHistories Batch

This batch process deletes UIN detail history records (UIN Audit Information). UIN Audit information could be purged for a UIN while the UIN is still open within the system. Open UIN is any UIN that is in one of the following statuses:

- In Stock
- In Receiving
- Reserved for Shipping
- Unavailable

UIN history records with open status and an update date at least *X* days in the past (where *X* is system parameter DAYS_TO_HOLD_UIN_AUDIT_INFORMATION) are deleted from ITEM_UIN_HISTORY table.

Usage

```
purgeUINDetailHistories.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

PurgeUserCache Batch

This batch program deletes expired user cache records which include expired user authentication caches and authorization cache records.

The cached user passwords with a cache date that is at least *X* hours in the past (where *X* is the system parameter SECURITY_USER_AUTHENTICATION_CACHE_HOURS) will be deleted.

The batch process also find users with a cache date that is at least *X* hours in the past (where *X* is the system parameter SECURITY_USER_AUTHORIZATION_CACHE_HOURS). All cached role and store assignments are deleted for these users.

If the users still have existing role assignments, store assignments, or passwords then the user's cache date is set to **null**.

If the user has no existing assignments or passwords then the user is deleted from AC_USER_PASSWORD, AC_USER_ROLE, AC_USER_STORE, and AC_USER tables.

Usage

```
PurgeUserCache.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

If purge_date is not specified, the current GMT time is used.

PurgeUserPasswordHistory Batch

This batch process finds users with more than *X* passwords (where *X* is the system parameter PASSWORD_NUMBER_OF_PREVIOUS_TO_DISALLOW), and deletes the oldest passwords that exceed this limit.

Usage

```
PurgeUserPasswordHistory.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

Purge_date currently has no effect, reserved for future use.

PurgeWHDReceivings Batch

This batch process deletes the Warehouse delivery receivings which are in Completed or Cancelled status. The warehouse receivings records which are older than the DAYS_TO_HOLD_RECEIVED_SHIPMENTS will get purged, based on the value set for this parameter.

Usage

```
PurgeWHDReceivings.sh <purge_date>
```

Where `purge_date` is optional and the date format must be `dd/MM/yyyy` if purge_date is specified.

A

Appendix: Batch File Layout Specifications

This chapter describes the batch file layout specification.

DexnexFileParser Import File Layout Specification

This section includes the import file layout specification.

File Structure – 894 Delivery

DEX/NEX uses the EDI Standard 894 Transaction Set to communicate with the direct delivery receiving system. The basic format for the file is as shown in Table A–1:

Table A–1 DexnexFileParser Batch File Structure

Header	
ST = Transaction Set Header	
G82 = Delivery/Return Base Record	
N9 = Reference Identification	
	Detail (repeating…)
	LS = Loop Header
	G83 = Line Item Detail DSD
	G72 = Allowance or Charge at Detail Level
	LE = Loop Trailer
Summary	
G84 = Delivery/Return Record Totals	
G86 = Signature	
G85 = Record Integrity Check	
SE = Transaction Set Trailer	

- ST – Contains the transaction set number (for example, 894) and a control number.
- G82 – Contains the type of delivery (Delivery or Return), supplier information, and delivery date.
- N9 – Contains additional supplier information (Canada only).
- LS – Contains an ID for the details loops to follow.
- G83 – Contains the item #, quantity, UOM, unit cost, and item description.
- G72 – Contains allowance (for example, 10% off) or charge (for example, environmental levy) information.

- LE – Contains the loop trailer.
- G84 – Contains the total quantity and cost of the delivery.
- G86 – Contains the suppliers UCC signature.
- G85 – Contains an authentication identifier.
- SE – Contains the number of transactions in the transmission.

Table A–2 provides details of the DexnexFileParser batch file:

Table A–2 DexnexFileParser Batch File Details

Segment	Sub-Segment	Name	Required?	SIM value
ST		Transaction Set Header	Yes	
ST	ST01	Transaction Set ID Code	Yes	894 - identifies the EDI file type, use to validate.
ST	ST02	Transaction Set Control #	Yes	Ignore
G82		Delivery/Return Base Record	Yes	
G82	G8201	Credit/Debit Flag Code	Yes	D=Delivery, C=Return.
G82	G8202	Supplier's Delivery/Return Number	Yes	Use as supplier's purchase order number.
G82	G8203	DUNS Number	Yes	Ignore
G82	G8204	Receiver's Location Number	Yes	Contains the Store #
G82	G8205	DUNS Number	Yes	Supplier's DUNS Number - use to determine supplier
G82	G8206	Supplier's Location Number	Yes	Supplier's DUNS Location - use with DUNS Number to determine supplier
G82	G8207	Delivery/Return Date	Yes	Delivery Date
N9		Reference Identification	No	
N9	N901	Reference Identifier Qualifier	Yes	Ignore
N9	N902	Reference Number	Yes	Use as SIM invoice number
N9	N903	Free-Form Description	No	Ignore
LS	LS01	Loop Header	Yes	Provides an ID for the loop to follow in the file
G83		Line Item Detail	Yes	
G83	G8301	DSD Number	Yes	Ignore
G83	G8302	Quantity	Yes	Unit Quantity
G83	G8303	Unit of Measure Code	Yes	CA = Case, EA = Each
G83	G8304	UPC Item Number		
G83	G8305	Product ID Qualifier		
G83	G8306	Product ID Number		
G83	G8307	UPC Case Code	No	Pack Number
G83	G8308	Item List Cost	No	Unit Cost
G83	G8309	Pack	No	
G83	G8310	Cash Register Description	No	Ignore
G72		Allowance or Charge at Detail Level	No	Ignore

Table A–2 (Cont.) DexnexFileParser Batch File Details

Segment	Sub-Segment	Name	Required?	SIM value
G72	G7201	Allowance or Charge Code		Ignore
G72	G7202	Allowance/Charge Handling Code		Ignore
G72	G7203	Allowance or Charge Number		Ignore
G72	G7205	Allowance/Charge Rate		Ignore
G72	G7206	Allowance/Charge Quantity		Ignore
G72	G7207	Unit of Measure Code		Ignore
G72	G7208	Allowance/Charge Total Amount		Ignore
G72	G7209	Allowance/Charge Percent		Ignore
G72	G7210	Dollar Basis for Allow/Charge %		Ignore
LE	LE01	Loop Identifier		Loop Trailer, will contain same ID as loop header
G84		Delivery/Return Record Totals	Yes	
G84	G8401	Quantity	Yes	Sum of all G8302 values
G84	G8402	Total Invoice Amount	Yes	Total Cost, inclusive of charges and net of allowances.
G86	G8601	Signature	Yes	Ignore
G85	G8501	Integrity Check Value	Yes	Ignore
SE	SE01	Number of Included Segments	Yes	Total # of segments between ST and SE, used for validation
SE	SE02	Transaction Set Control #	Yes	Same as ST02, used for validation
GE	GE01	Number of transaction sets included	Yes	# of sets in functional group, used for validation
GE	GE02	Group Control Number	Yes	Same as GS06, used for validation

ThirdPartyStockCountParser Import File Layout Specification

Third Party Stock Count Import File Format:

Pipe-delimited (|) file contains store count data for a store and stock count ID as shown in Table A–3:

Table A–3 Third Party Stock Count Import File

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Store Number	Char(10)		Store number file was uploaded for. It is assumed only one store is passed in per file. (Required)
	Stock Count ID	Number(12)		Unique identifier for item. Assumption is SIM will always take first stock count ID listed. (Required)
FDETL	Record Descriptor	Char(5)	FDETL	Detail record marker.
	Stock Count Date	Date(14)		Indicates date/time item was physically counted by the third party. (*YYYYMMDDHH24MISS*) For example, 20091019134600 (Required) **Note:** If not using timestamp, use **00** for time.
	Area Number	Char(10)		10-digit code indicating where in the store the item is located. (Optional)
	UPC or Item Number	Char(25)		25-digit universal product code. (Required)
	Count Quantity	Number (12,4)		Quantity counted for item, required. This field must allow for decimals when counting in UOM other than each. (Required)
	UIN(Item Serial Number)	Char(128)		Unique identification serial number for item, required if current item requires serial number.
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker.

The following is a sample Third Party Stock Count Import File:

```
FHEAD|5000|1074|
FDETL|20091014235959|1|100665085|1|ItemSerialNum1234|
FDETL|20091014140000|1|100665085|1|ItemSerialNum9999|
FDETL|20091014000000|1|100665085|1||
FTAIL|
```

ClearancePriceChange Import File Layout Specification

Table A–4 ClearancePriceChange File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Line ID	Number(10)	1	Unique line ID
	File Type	Char(5)	CLRPC	Clearance Price Changes
	Export timestamp	Timestamp		System clock timestamp (YYYYMMDDHHMISS)
	Format Version	Char(5)	1.0	File Format Version
FDETL	Record Descriptor	Char(5)	FDETL	File Detail Marker (one per clearance create or modify)
	Line ID	Number(10)		Unique line ID
	Event Type	Char(3)		**CRE** = Create **MOD** = Modify
	ID	Number(15)		Clearance identifier
	Item	Char(25)		Item identifier
	Location	Number(10)		Location identifier
	Location Type	Char(1)		**S** = Store **W** = Warehouse
	Effective Date	Date		Clearance Effective Date (YYYMMDDHHMISS)
	Selling Retail	Number(20,4)		Selling retail with price change applied
	Selling Retail UOM	Char(4)		Selling retail unit of measure
	Selling Retail Currency	Char(3)		Selling retail currency
	Reset Clearance ID	Number(15)		ID of clearance reset
FDELE	Record Descriptor	Char(5)	FDELE	File Detail Delete Marker (one per clearance delete)
	Line ID	Number(10)		Unique line ID
	ID	Number(15)		Clearance identifier
	Item	Char(25)		Item identifier
	Location	Number(10)		Location identifier
	Location Type	Char(1)		**S** = Store **W** = Warehouse
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker
	Line ID	Number(10)		Unique line ID
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

RegularPriceChange Import File Layout Specification

Table A–5 RegularPriceChange File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Line ID	Number(10)	1	Unique line ID
	File Type	Char(5)	REGPC	Regular Price Changes
	Export timestamp	Timestamp		System clock timestamp (YYYYMMDDHHMISS)
	Format Version	Char(5)	1.0	File Format Version
FDETL	Record Descriptor	Char(5)	FDETL	File Detail Marker (one per price change create or modify)
	Line ID	Number(10)		Unique line ID
	Event Type	Char(3)		**CRE** = Create **MOD** = Modify
	ID	Number(15)		Price Change identifier
	Item	Char(25)		Item identifier
	Location	Number(10)		Location identifier
	Location Type	Char(1)		**S** = Store **W** = Warehouse
	Effective Date	Date		Effective Date of price change (YYYYMMDDHHMISS)
	Selling Unit Change Ind	Number(1)		Did selling unit retail change with this price event (**0** = no change, **1** = changed)
	Selling Retail	Number(20,4)		Selling retail with price change applied
	Selling Retail UOM	Char(4)		Selling retail unit of measure
	Selling Retail Currency	Char(3)		Selling retail currency
	Multi-Unit Change Ind	Number(1)		Did multi unit retail change with this price event (**0** = no change, **1** = changed)
	Multi-Units	Number(12,4)		Number Multi Units
	Multi-Unit Retail	Number(20,4)		Multi Unit Retail

Table A–5 (Cont.) RegularPriceChange File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Multi-Unit UOM	Char(4)		Multi Unit Retail Unit Of Measure
	Multi-Unit Currency	Char(3)		Multi Unit Retail Currency
FDELE	Record Descriptor	Char(5)	FDELE	File Detail Delete Marker (one per price change delete)
	Line ID	Number(10)		Unique line ID
	ID	Number(15)		Price Change identifier
	Item	Char(25)		Item identifier
	Location	Number(10)		Location identifier
	Location Type	Char(1)		**S** = Store **W**= Warehouse
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker
	Line ID	Number(10)		Unique line ID
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

PromotionPriceChange Import File Layout Specification

Table A–6 PromotionPriceChange Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Line ID	Number(10)	1	Unique line Identifier
	File Type	Char(5)	PROMO	Promotions
	Export timestamp	Timestamp		System clock timestamp (YYYYMMDDHHMISS)
	Format Version	Char(5)	1.0	File Format Version
TMBPE	Record Descriptor	Char(5)	TMBPE	Promotion (transaction head)
	Line ID	Number(10)		Unique line identifier
	Event Type	Char(3)		**CRE** = Create **MOD**= Modify
TPDTL	Record Descriptor	Char(5)	TPDTL	Promotion Detail Component
	Line ID	Number(10)		Unique line identifier
	Promo ID	Number(10)		Promotion identifier
	Promo Comp ID	Number(10)		Promotion Component ID
	Promo Name	Char(160)		Promotion Header Name

Table A–6 (Cont.) PromotionPriceChange Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Promo Desc	Char(640)		Promotion Header Description
	Promo Comp Desc	Char(160)		Promotion Component Name
	Promo Type	Number(2)		Valid values will be modified to be as follows: 0 - Multi-Buy Promotion; 1 - Simple Promotion; 2 - Threshold Promotion; 3 - Finance Promotion (formerly tied to a value of 6); 4 - Transaction Promotion
	Promo Comp Detail ID	Number(10)		Promotion Component Detail identifier
	Start Date	Date		Start Date of Promotion Component Detail (YYYYMMDDHH24MISS)
	End Date	Date		End Date of Promotion Component Detail (YYYYMMDDHH24MISS)
	Apply To Code	Number(1)		Holds the apply to code for the promotion detail. Determines if the promotion is applied to regular retail only (no clearances in effect), clearance retail only (only when a clearance is in effect) or both regular and clearance retail. Valid values are 0 - Regular Only; 1 - Clearance Only; 2 - Regular and Clearance.
	Discount Limit	Number(3)		The number of times that the promotion that can be applied to a transaction.
	Apply Order	Number(1)		Application Order of the Promotion
	Threshold ID	Number(6)		Threshold identifier
	Customer Type ID	Number(10)		Customer Type identifier
	Threshold Qualification Type	Number(1)		The qualification type for the threshold. Will only be populated for threshold promotions. Valid values are 0 for item level and 1 for threshold level.
TLLST	Record Descriptor	Char(5)	TLLST	Promotion Detail Component
	Line ID	Number(10)		Unique line identifier
	Location ID	Number(10)		Org Node [Store or Warehouse] identifier
	Location Type	Char(1)		Org Node Type [Store or Warehouse]
TPGRP	Record Descriptor	Char(5)	TPGRP	Promotion Detail Group
	Line ID	Number(10)		Unique line identifier
	Group ID	Number(10)		Group Number
TGLIST	Record Descriptor	Char(5)	TGLIST	Promotion Group List

Table A–6 (Cont.) PromotionPriceChange Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Line ID	Number(10)		Unique line identifier
	List ID	Number(10)		List identifier
	Reward Application	Number(1)		How this reward is applied to the promotion detail.
	Description	Char(120)		Description
	Price Range Min	Number (20,4)		Contains price range promotion minimum valid retail value.
	Price Range Max	Number (20,4)		Contains price range promotion maximum valid retail value.
TLITM	Record Descriptor	Char(5)	TLITM	Promotion Group List
	Line ID	Number(10)		Unique line identifier
	Item ID	Char(25)		Transaction Item Identifier
TPDSC	Record Descriptor	Char(5)	TPDSC	Discount Detail for List
	Line ID	Number(10)		Unique line identifier
	Change Type	Number(2)		Change Type
	Change Amount	Number(20,4)		Change Amount
	Change Currency	Char(3)		Change Currency
	Change Percent	Number(20,4)		Change Percent
	Change Selling UOM	Char(4)		Change Selling UOM
	Qual Type	Number(2)		Qualification Type
	Qual Value	Number(2)		Qualification Value
	Change Duration	Number(20,4)		Change Duration
TPILSR	Record Descriptor	Char(5)	TPILSR	Items in Promotion
	Line ID	Number(10)		Unique line identifier
	Item ID	Char(25)	TTAIL	Transaction Item Identifier
	Selling Retail	Number(20,4)		Selling retail of the item
	Selling UOM	Char(4)		Selling UOM of the item
	Location ID	Number(10)		Org Node [Store or Warehouse] identifier
	Effective Date	Date		Effective Date of the selling retail - YYYYMMDDHH24MISS
	Selling Retail Currency	Char(3)		Selling retail currency
TPCDT	Record Descriptor	Char(5)	TPCDT	Credit Detail
	Credit Detail ID	Number(10)		Credit Detail ID
	Line ID	Number(10)		Unique line id
	Credit Type	Char(40)		Credit Type
	binNumberFrom	Number(10)		BinNumber From
	binNumberTo	Number(10)		Bin Number To
	Commission Rate	Number(10)		Commission Rate
	Comments	Char(160)		Comments

Table A–6 (Cont.) PromotionPriceChange Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
TTAIL	Record Descriptor	Char(5)	TTAIL	Transaction Tail
	Line ID	Number(10)		Unique line identifier
TPCIL	Record Descriptor	Char(5)	TPCIL	Cancel Item Loc
	Line ID	Number(10)		Unique line identifier
	Promo ID	Number(10)		The ID of the promotion
	Promo Comp Id	Number(10)		Promotion Component Id
	Promo Comp Detail Id	Number(10)		Promotion Component Detail identifier
	Item Id	Char(25)		Transaction Item Identifier for item
	Location Id	Number(10)		Org Node (Store or Warehouse) identifier
	Location Type	Char(1)		Org Node Type (Store or Warehouse). Valid values are 'S' and 'W'
	Cancellation Date	Date		Cancellation effective date - YYYYMMDDHH24MISS
FPDEL	Record Descriptor	Char(5)	FPDEL	Delete Promotion
	Line ID	Number(10)		Unique line identifier
	Promo ID	Number(10)		The ID of the promotion
	Promo Comp Id	Number(10)		Promotion Component Id
	Promo Comp Detail ID	Number(10)		Promotion Component Detail ID
	Group ID	Number(10)		Group Number
	List ID	Number(10)		List ID
	Item ID	Char(25)		Transaction Item Identifier for item
	Location ID	Number(10)		Org Node [Store or Warehouse] identifier
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker
	Line ID	Number(10)		Unique line identifier
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

POS Sale Transaction Import File (SIMT-LOG) Specification

The input file would be in Pipe ('|') delimited format.

Table A–7 SIMT-LOG file

Record Name	Field Name	Field Type	Default Value	Description
FILE HEADER	FILE HEADERFile Type Record Descriptor	VARCHAR2(5)	FHEAD	Identifies the File Record Type
	Location Number	NUMBER(10)		Store Number
	Business Date	VARCHAR2(14)		Business Date of transactions in YYYYMMDDHHSS format

Table A–7 (Cont.) SIMT-LOG file

Record Name	Field Name	Field Type	Default Value	Description
	File Creation Date	VARCHAR2(14)	SYSDATE	File Create Date in YYYMMDDHHMSS format
TRANSACTION HEADER	File Type Record Descriptor	VARCHAR2 (5)	THEAD	Identifies the File Record Type
	Transaction Number	VARCHAR2(128)		The unique transaction reference number generated by POS/OMS.
	Transaction Date and Time	VARCHAR2(14)		Date transactions were processed in POS/OMS
	Customer Order Id	VARCHAR2(128)		External customer order id, if transaction is a customer order
	Customer Order Comments	VARCHAR(512)		Comments on the customer order
TRANSACTION DETAIL	File Type Record Descriptor	VARCHAR2(5)	TDETL	Identifies the File Record Type
	Item Id	VARCHAR2(25)		ID number of the item.
	UIN	VARCHAR2(128)		This is the UNIQUE_ID value from RTLOG
	Item Quantity	NUMBER(12,4)		Quantity of the item on this transaction
	Selling UOM	VARCHAR2(4)		UOM at which this item was sold
	Reason Code	NUMBER(4)		Reason entered by cashier for some transaction types. Required for voids, returns, for example.
	Comments	VARCHAR(512)		Comments for this line item
	Transaction Code	VARCHAR2(25)		The type of sale represented by this line item. Valid value are SALE,RETURN,VOID_ SALE,VOID_RETURN,ORDER_ NEW,ORDER_FULFILL,ORDER_ CANCEL,ORDER_CANCEL_ FULFILL
	Reservation Type	VARCHAR(25)		Reservation type if POS transaction is a customer order. Valid values are SPECIAL_ORDER, WEB_ ORDER, PICKUP_AND DELIVERY,LAYAWAY
	Fulfillment Order Number	VARCHAR2(48)		Fulfillment Order Number from OMS
	Drop Ship Indicator	VARCHAR(1)		'P' if it is drop ship otherwise 'N'
TRANSACTION TAIL	File Record Type Descriptor	VARCHAR2(5)	TTAIL	Identifies the File Record Type
	Transaction Record Counter	NUMBER(6)		Number of TDETL records in this transaction set.
FILE TAIL	File Record Type Descriptor	VARCHAR2(5)	FTAIL	Identifies the File Record Type
	File Record Counter	NUMBER(10)		Number of records/transactions processed in current file (only records between head and tail)

RetailSaleAuditImport SIM-ReSA File Specification

The input file would be in Pipe ('|') delimited format.

Table A–8 Stock Count Export File Layout

Record Name	Field Name	Field Type	Default Value	Description
FILE HEADER	FILE Type Record Descriptor	VARCHAR2(5)	FHEAD	Identifies the File Record Type
	File Line Id	VARCHAR(10)		Sequential file line number
	File Type Definition	VARCHAR2(4)	SIMT	Identifies the File Type
	Location Number	NUMBER(10)		Store Number
	Business Date	VARCHAR2(14)		Business Date of transactions in YYYYMMDDHHSS format
	File Creation Date	VARCHAR2(14)	SYSDATE	File Create Date in YYYMMDDHHMSS format
TRANSACTION HEADER	File Type Record Descriptor	VARCHAR2 (5)	THEAD	Identifies the File Record Type
	File Line Id	VARCHAR(10)		Sequential file line number
	Transaction Number	NUMBER(10)		The unique transaction reference number generated by POS/OMS
	Revision Number	NUMBER(3)		The version of the transaction being sent
	Transaction Date and Time	VARCHAR2(14)		Date transactions were processed in POS/OMS
	Transaction Type	VARCHAR2(14)		Transaction Type Code (for example, SALE, RETURN, SPLORD)
	Pos created flag	VARCHAR2(1)		'Y' identifies that the transaction occurred at POS, 'N' identifies that the transaction was created in ReSA
TRANSACTION DETAIL	File Type Record Descriptor	VARCHAR2(5)	TDETL	Identifies the File Record Type
	File Line ID	VARCHAR(10)		Sequential file line number.
	Item Sequence Number	NUMBER(4)		The order in which items were entered during a transaction
	Item	VARCHAR2(25)		ID number of the item.
	Item Type	VARCHAR2(6)		Type of Item sold. Can be 'ITEM', 'REF', 'GCN', 'NMITEM'
	Reference Item	VARCHAR2(25)		Identified sub-transaction level merchandise item.
	Non-Merchandise Item	VARCHAR2(25)		Identifies non-merchandise item
	Item Status	VARCHAR2(6)		Status of the item within the transaction, V for item void, S for sold item, R for returned item, Layaway Initiate (LIN), Layaway Cancel, Layaway Complete (LCO), Order Initiate (ORI), Order Cancel (ORC) Order Complete (ORD)
	Serial Number	VARCHAR2(128)		This is the UNIQUE_ID value from RTLOG

Table A–8 (Cont.) Stock Count Export File Layout

Record Name	Field Name	Field Type	Default Value	Description
	Pack Indicator	VARCHAR2(1)		Pack indicator of item sold or returned
	Catch Weight Indicator	VARCHAR2(1)		Indicates if item is a catchweight item
	Item Quantity Sign	VARCHAR2(1)		Determines if the Total Sale Quantity is positive or negative 'P' - Positive 'N' - Negative
	Item Quantity Value	NUMBER(20)		Total sales value of goods sold/returned (4 implied decimal places), for example, Total Quantity * 10000
	Standard UOM	VARCHAR2(4)		Standard UOM of the Item
	Selling UOM	VARCHAR2(4)		UOM at which this item was sold
	Wastage Type	VARCHAR2(6)		Wastage type of item sold or returned
	Wastage Percentage	NUMBER(12)		Wastage Percent*10000 (4 implied decimal places), wastage percent of item sold or returned
	Drop Ship Indicator	VARCHAR2(1)	N	This will always be N for SIM Export
	Actual Weight Quantity	NUMBER(12)		Actual Weight Quantity*10000 (4 implied decimal places), the actual weight of the item, only populated if catchweight_ind = 'Y'
	Reason Code	VARCHAR2(6)		Reason entered by cashier for some transaction types
	Sale Value	NUMBER(20)		Total Sales Value * 10000 (4 implied decimal places), sales value, net sales value of goods sold
	Sales Sign	VARCHAR2(1)		Determines if the Total Sales Value is positive or negative 'P' - Positive 'N' - Negative
	Unit Retail	NUMBER(20,4)		Unit retail with 4 implied decimal places
	Sales Type	VARCHAR2(1)		Indicates if the line item is a Regular Sale, a CO serviced by OMS (External CO), or a CO serviced by SIM (In-Store CO)
	Customer Order Number	VARCHAR2(50)		Customer Order Number
	Customer Order Type			Customer order type
	Fulfillment Order Number	VARCHAR2(50)		Fulfillment Order Number from OMS
TRANSACTION TAIL	File Record Type Descriptor	VARCHAR2(5)	TTAIL	Identifies the File Record Type

Table A–8 (Cont.) Stock Count Export File Layout

Record Name	Field Name	Field Type	Default Value	Description
	File Line Id	NUMBER(10)		Sequential file line number
	Transaction Record Counter	NUMBER(6)		Number of TDETL records in this transaction set
FILE TAIL	File Record Type Descriptor	VARCHAR2(5)	FTAIL	Identifies the File Record Type
	File Line Id	NUMBER(10)		Sequential file line number
	File Record Counter	NUMBER(10)		Number of records/transactions processed in current file (only records between head and tail)

UINAttributeImport File Specification

The input file will have the following comma-separated values:

Table A–9 UIN Attribute File

Record Name	Field Name	FieldType	Default Value	Description
1	Store ID	Number	NA	ID of the Store, for example, **1111**
2	Department ID	Number	NA	ID of the Department, for example, **2222**
3	Class ID	Number	NA	ID of the Class, for example, **2**
4	Type	Varchar2	NA	Type of UIN, for example, **Serial Number**
5	Label	Varchar2	NA	Label of UIN, for example, **Serial Number**
6	Capture_Time_ID	Number	NA	Capture Time ID, for example, **2**
7	External_Create_ Allowed	Varchar2	NA	External Create Allowed or not, for example, **Y** or **N**

The following is a sample input file:

```
3000,3023,2,'Auto Generate SN','Serial Number',2,False
```

Stock Count Results Export File Specification

The stock count result export file is generated when unit amount stock count authorization completes. The stock count authorization process can be a manual authorization or invoked by third party stock count batch for an auto-authorized unit amount stock count. This export file can be uploaded to RMS by RMS file to update their inventory with the actual physical stock count.

Table A–10 SIMT-ReSA File

Record Name	Field Name	Field Type	Description
File Header	file type record descriptor	Char(5)	hardcode FHEAD
	file line identifier	Number(10)	ID of current line being processed, hardcode **000000001**
	file type	Char(4)	hardcode STKU
	file create date	Date(14)YYYYMMDDHHMISS	date written by convert program
	stocktake_date	Date(14)YYYYMMDDHHMISS	take_head.stocktake_date
	cycle count	Number(8)	stake_head.cycle_count
	loc_type	Char(1)	hardcode **W** or **S**
	location	Number(10)	stake_location.wh or stake_location.store
Transaction record	file type record descriptor	Char(5)	hardcode **FDETL**
	file line identifier	Number(10)	ID of current line being processed, internally incremented
	item type	Char(3)	hardcode **ITM**
	item value	Char(25)	item ID
	inventory quantity	Number(12,4)	total units or total weight
	location description	Char(30)	Where in the location the item exists. For example, Back Stockroom or Front Window Display
File trailer	file type record descriptor	Char(5)	hardcode **FTAIL**
	file line identifier	Number(10)	ID of current line being processed, internally incremented
	file record count	Number(10)	Number of detail records

StoreSequenceDataParser Import File Layout Specification

The input file would be in pipe ('|') delimited format.

Table A–11 StoreSequenceImport File

Record Name	Field Name	FieldType	Description
File Header	file type record descriptor	Char(5)	hardcode FHEAD
	Store ID	Number(10)	Store identifier
Sequence record	file type record descriptor	Char(5)	hardcode SHEAD
	Area type	Number(9)	The Store Sequence Area. 0 = None, 1 = Shopfloor, 2 = Backroom
	Child sequenced	Varchar2(1)	'Y' if child is sequenced, 'N' if not
	Department ID	Number(12)	Department ID
	Class ID	Number(12)	Class ID
	Description	Varchar2(255)	Description of Store Sequence

Table A–11 (Cont.) StoreSequenceImport File

Record Name	Field Name	FieldType	Description
	Not sequenced	Varchar2(1)	The order the store sequence is in compared to other store sequences
	Sequence Order	Number(20)	Y indicates a default sequence containing all items that have not been sequenced elsewhere
Sequence detail	file type record descriptor	Char(5)	hardcode SDETL
	Item ID	Varchar2(25)	Item ID
	Primary location	Varchar2(1)	Indicator if the location specified is the primary location for the item, Y if is primary location for item, N otherwise
	Item sequence order	Number(20)	Order of item within store sequence
	Capacity	Number(11,2)	The size of the location appropriate to unit of measure
	Ticket quantity	Number(11,2)	The quantity of tickets that need to be printed or used for the item inventory location
	Ticket format ID	Number(10)	Item ticket format identifier
	Uom mode	Number(2)	The Unit Of Measure display mode: 1 = Units, 2 = Cases
	Width	Number(12)	Width value to indicate how many items can fit across the width of the shelf
Sequence trailer	File type record descriptor	Char(5)	hardcode STAIL
File trailer	File type record descriptor	Char(5)	hardcode FTAIL

B

Appendix: Setup Auto-Authorized Third-Party Stock Count

This section describe steps to setup an auto authorize third party stock count:

1. Set up a product group with counting method as **Third Party** and with auto-authorize flag checked.
2. Create a new product group schedule on the Product Group screen.
3. Run the ExtractUnitAmountStockCount.sh batch program and ExtractUnitStockCount.sh batch to generate the stock counts.

   Note: After the batch has completed, from the Main Menu go to Inv Mgmt>Stock Counts>Stock Count List screen. Notice that a separate stock count record has been created for each department. The batch creates stock count groups for all items for all departments for the store, including items with SOH values of zero grouped by department. For each department record, the Stock Count Type and Status from the stock count list screen will be **Type = Stock Count** and **Status = New**.

4. Take a snapshot of the SOH on Stock Count List screen.

   The snapshot must be taken before uploading the third-party flat file.

   Note: Selecting **Take Snapshot** takes a snapshot of the current SIM SOH figure, and assigns this to every item in the stock count records. The snapshot button is displayed only if there is an extracted Third Party Stock Count or Unit and Amount stock count on the Stock Count List screen. You must first select at least one record from the Third Party Stock Count in order for the snapshot to be taken. Status of the stock count will change to In Progress. This will indicate that the snapshot has occurred. The user will not be able to access the stock count records until the file has been uploaded. If the user double-clicks one of the department stock counts on the list screen, SIM will prompt with the message **The stock count will not be accessible until the import process has completed**. The user will not be able to drill into the detail screen if the third-party file has not yet been imported into SIM.

5. Upload third-party count file to SIM:

Once counting is complete, run ThirdPartyStockcountImport batch to process the data file.

See the ThirdPartyStockCountImport Batch for details.

C

Appendix: SIM Integration Connection Troubleshooting

This appendix describes some common SIM integration connection problems, and identifies some common remedies:

SIM Message Publishing

You can try the following if you are experiencing issues with the SIM message publishing:

- Verify RIB-SIM server is running.
- Verify the remote_service_locator_info_ribclient.xml file in SIM server points to the correct RIB-SIM server.
- The user name and password for the wallet alias used in SIM server remote_service_locator_info_ribclient.xml file exists in RIB-SIM application-managed server.

SIM Message Subscribing

You can try the following if you are experiencing issues with the SIM message subscribing:

- Verify SIM server is running
- Verify RIB-SIM server remote_service_locator_info_ribclient.xml is point to the correct SIM server
- Verify the user name and password for the wallet alias used in RIB-SIM server remote_service_locator_info_ribclient.xml exists in SIM application managed server.

See the Oracle Retail Integration Bus documentation for detailed troubleshooting information.