Retek® Store Inventory Management™ (SIM) 10.3

Operations Guide

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403
USA

888.61.RETEK (toll free US)
Switchboard:
+1 612 587 5000

Fax:
+1 612 587 5100

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom

Switchboard:
+44 (0)20 7563 4600

Sales Enquiries:
+44 (0)20 7563 46 46

Fax:
+44 (0)20 7563 46 10

The software described in this documentation is furnished under a license agreement, is the confidential information of Retek Inc., and may be used only in accordance with the terms of the agreement.

No part of this documentation may be reproduced or transmitted in any form or by any means without the express written permission of Retek Inc., Retek on the Mall, 950 Nicollet Mall, Minneapolis, MN 55403, and the copyright notice may not be removed without the consent of Retek Inc.

Information in this documentation is subject to change without notice.

Retek provides product documentation in a read-only-format to ensure content integrity. Retek Customer Support cannot support documentation that has been changed without Retek authorization.

Retek® Store Inventory Management™ is a trademark of Retek Inc.

Retek and the Retek logo are registered trademarks of Retek Inc.

This unpublished work is protected by confidentiality agreement, and by trade secret, copyright, and other laws. In the event of publication, the following notice shall apply:

©2003 Retek Inc. All rights reserved.

All other product names mentioned are trademarks or registered trademarks of their respective owners and should be treated as such.

Printed in the United States of America.

Customer Support

Customer Support hours

Customer Support is available 7x24x365 via email, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance. Retek customers on active maintenance agreements may contact a global Customer Support representative in accordance with contract terms in one of the following ways.

Contact Method	Contact Information
E-mail	support@retek.com
Internet (ROCS)	rocs.retek.com Retek's secure client Web site to update and view issues
Phone	1 612 587 5800

Toll free alternatives are also available in various regions of the world:

Australia	1 800 555 923 (AU-Telstra) or 1 800 000 562 (AU-Optus)
France	0800 90 91 66
United Kingdom	0800 917 2863
United States	1 800 61 RETEK or 800 617 3835
Mail	Retek Customer Support Retek on the Mall 950 Nicollet Mall Minneapolis, MN 55403

When contacting Customer Support, please provide:

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

Contents

.

Chapter 1 – Introduction

This operations guide serves as a Retek Store Inventory Management (SIM) reference to explain 'backend' processes. SIM is designed as a standalone application that can be customized to work with any merchandising system.

Overview

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 performs 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.

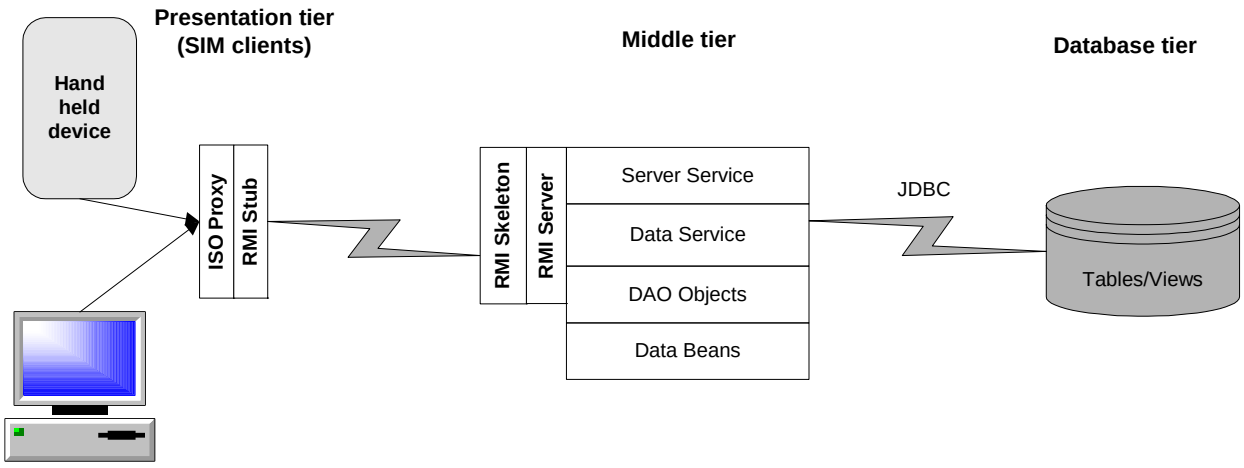
Technical architecture overview

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 back-end system. The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify. Any given tier need not be concerned with the internal functional tasks of any other tier.

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. The server is not deployed within the store. The application's clients talk to the server across the wire in almost real time.

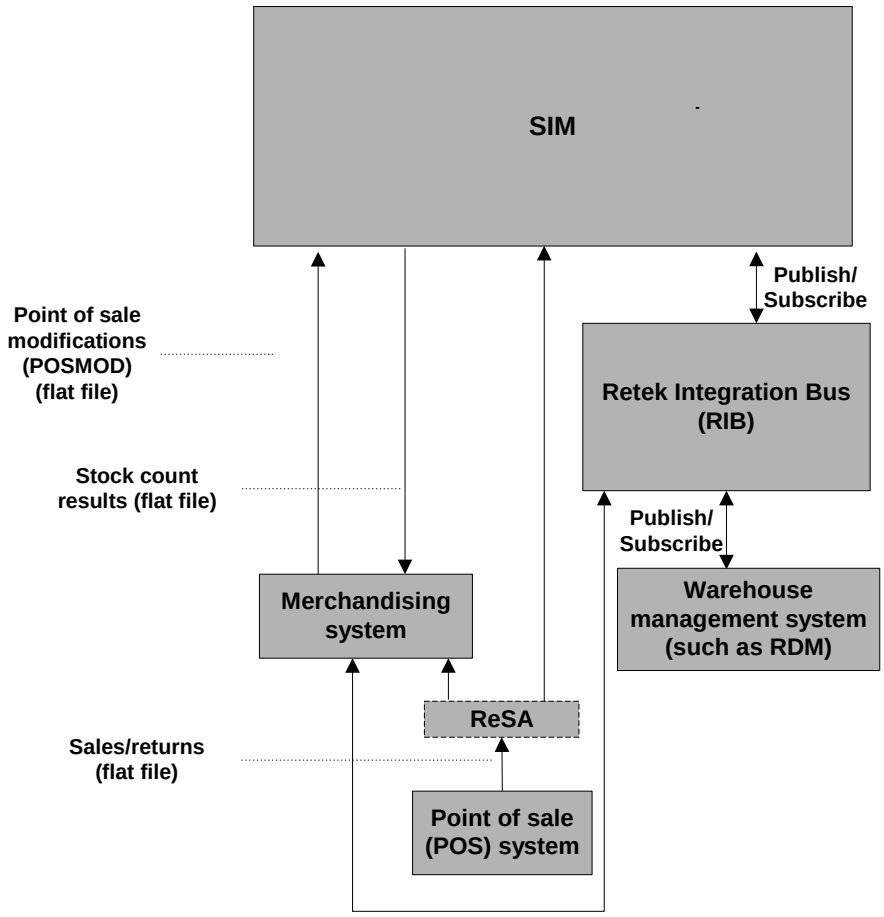
The following diagram offers a high-level conceptual view of the tiers. For a detailed description of this diagram, see Chapter 3, "Technical architecture".



SIM's technical architecture

SIM's integration points into the retail enterprise

The following high-level diagram shows the overall direction of the data among systems and products across the enterprise. For a detailed description of this diagram, see Chapter 5, "Integration interface dataflows".



SIM-related dataflow across the enterprise

Who this guide is written for

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

- System analysts and system operation personnel:
  - who are looking for information about SIM's processes internally or in relation to the systems across the enterprise.
  - who operate SIM on a regular basis.
- Integrators and implementation staff who have the overall responsibility for implementing SIM into their enterprise.
- Business analysts who are looking for information about processes and interfaces to validate the support for business scenarios within SIM and other systems across the enterprise.

Javadoc for SIM

Javadoc is the tool from Sun Microsystems that generates API documentation in HTML format. Retek provides Javadoc documentation generated from SIM code as a separate documentation deliverable (along with the User Guide, Application Installation Guide, and so on). Click the HTML file named 'index' in the applicable Javadoc folder to open the Javadoc.

Where you can find more information

- SIM front-end documentation (for example, the SIM User Guide)
- SIM Installation Guide
- Retek Distribution Management (RDM) product documentation
- Retek Merchandising System (RMS) product documentation
- Retek Integration Guide and other RIB-related documentation

Chapter 2 – 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.

Configuration (.cfg) files

Client-defined configurations for SIM are located in .cfg files. The key system parameters contained in these file are described in this section.

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

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

JDBC configuration file (jdbc.cfg)

This file delineates how the system uses the persistence layer. Key SIM-related values within the file are shown below. Note that some values in the file may be intended for development purposes only or be related to another product (a POS product for example).

Oracle configuration

The configuration in this file instructs the system about the database in which the SIM tables reside. Note that a retailer using a specific database comments out any other database parameters in the configuration file. Under normal circumstances, the driver value should *not* change. The retailer establishes the machine name, the database name, the user name, and the password. The pool size pertains to the number of available database connections that the retailer intends to keep available in the pool. The application automatically adjusts the pool size as needed. JDBC_VERBOSE setting is for debugging purposes only. If a database connection is unused for a certain amount of time, it stops automatically. The lock setting describes how long the system attempts to procure a lock for the user before giving up and returning a message that the lock could not be attained.

For example:

```
# Oracle configuration
DATABASE=ORACLE
DRIVER=oracle.jdbc.driver.OracleDriver
URL=jdbc:oracle:thin:@<SIMDBMachineName>:1521:<SIMDBName>
POOL_INITIAL_SIZE=5
USER_NAME=<SIMUserId>
PASSWORD=<password>
JDBC_VERBOSE=false
#Time in seconds to keep unused connections (3600 = 1 hour)
CONNECTION_EXPIRATION=3600
#Time in seconds to wait on database locks
LOCK_WAIT=5
```

RMS Oracle configuration

This section is only valid if the retailer is using RMS. These parameters are used by the dataseeding program.

For example,

```
# RMS Oracle configuration
RMS_DATABASE=ORACLE
RMS_DRIVER=oracle.jdbc.driver.OracleDriver
RMS_URL=jdbc:oracle:thin:@<RMSDBMachineName>:1521:<RMSDBName>
RMS_POOL_INITIAL_SIZE=5
RMS_USER_NAME=<RMSUserId>
RMS_PASSWORD=<password>
RMS_JDBC_VERBOSE=false
#Time in seconds to keep unused connections (3600 = 1 hour)
RMS_CONNECTION_EXPIRATION=3600
```

Specific data access objects (DAO) implementations

These settings relate to data access-related information. The values allow the retailer to customize what class is used for data access for a given service. The settings in this section are (with few exceptions) associated with Oracle. The values should not to be changed if the system was purchased with RMS, which must use Oracle. A retailer doing custom work, however, can write an implementation that works for whatever database is being used and change the values in this section accordingly.

For example:

```
#Specific DAO Implementations

EMPLOYEE_DAO=com.chelseasystems.cs.dataaccess.rss.ldap.dao.EmployeeLdapDAO

LOCALITEM_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.LocalItemRSSOracleDAO

ALLOCATION_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.AllocationRSSOracleDAO

PACKITEM_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.PackItemRSSOracleDAO

ITEMMASTER_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.ItemMasterRSSOracleDAO

SOURCE_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.SourceRSSOracleDAO

MDSE_HIERARCHY_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.MdseHierarchyRSSOracleDAO

#LOCATION_DAO=com.chelseasystems.cs.dataaccess.rss.oracle.dao.StoreRSSOracleDAO
```

Network configuration file (network.cfg)

Connectivity between the SIM client and the middle tier is achieved via a Remote Naming Service (RNS), which provides the SIM client with the necessary IP address and port to access the SIM container. This configuration file includes parameters related to the naming service of Integrated Store Operations (ISO). The client sets the IP address to the server on which the code is running. The port number (40000 in the example below) can be left as the default, or it can be changed.

Note: If the retailer changes the port number, the equivalent value must also be changed in RNS-related .bat/.sh files. The values must be kept in sync.

The MAXIMUM_CONNECT_ATTEMPTS value tells the system how many times it should attempt to connect before moving into offline mode, which prevents the user from continuing to use the system. The VERSION setting should not have to be changed.

```
#  This is a comma-delimited list of addresses to the master naming
#  service.

#  Example:  10.9.1.47:3000,10.9.1.47:4000,10.9.1.47:5000

#MASTER_NODE=<NamingServerIpAddress>:40000

MASTER_NODE=10.6.1.162:40000


#  This is the address to the naming service for a specific
#  application server.

#  Each application server will have a unique address.

#  The Node Monitor application  will bind to this.

#APPLICATION_NODE=<AppServerIpAddress>:40001

APPLICATION_NODE=10.6.1.162:40001


#  Maximum number of times to try to bind to a naming service before
#  throwing a DowntimeException "Unable to find Naming Service".

MAX_CONNECT_ATTEMPTS=2


#  Version control number of the naming service that any client
#  should look for.  This enables multiple versions to exist
#  simultaneously.

VERSION=1
```

LDAP configuration file (ldap.cfg)

Security configuration is accomplished in the ldap.cfg file. The entries in this file point SIM to the appropriate machine, port, and so on to find the LDAP server. Key values within the file are shown below. Note that some values in the file may be intended for development purposes only or be related to another product (a POS product for example).

URL and port number for the LDAP server

If more than one LDAP server is being used, this entry can contain the primary server, secondary server, and so on. The retailer establishes the server name or IP address.

For example:

```
#URL and port number for the ldap server

#PRIMARY_LDAP_URL=ldap://<LDAPServerName or IP Address>:389

PRIMARY_LDAP_URL =ldap://10.6.1.21:389

BACKUP_LDAP_URL_1=ldap://10.6.1.21:389

BACKUP_LDAP_URL_2=ldap://10.6.1.21:389
```

LDAP schema details

The values in this entry must correspond to entries in the LDAP server. DN stands for domain name. CN stands for company name.

For example:

```
#LDAP Schema Details
BASE_DN=dc=retek,dc=com
#APPLICATION_LOGIN=cn=<LDAPuserLoginName>,dc=retek,dc=com
APPLICATION_LOGIN=cn=Retek,dc=retek,dc=com
#APPLICATION_PASSWORD=<LDAPuserLoginPassword>
APPLICATION_PASSWORD=retek
```

User data context

The values in this entry must correspond to entries in the LDAP server. Note that the value for the retekAppName is ‘rss’. The SUB_CONTEXT value exists to accommodate more than one application within a company.

For example:

```
#User Data Context
SUB_CONTEXT=retekAppName=rss,cn=Retek
RETEK_APP_NAME=rss
#CN=<LDAPuserLoginName>
CN=Retek
```

Clientmaster configuration file (clientmaster.cfg)

This file contains information related specifically to the SIM client. Note that some values in the file may be intended for development purposes only or be related to another product (a POS product for example).

Initial class that is run

These parameters are associated with initial system processing. Under normal operating conditions, these parameters should *not* be changed.

For example:

```
#  This is always the initial class that is run.  This class could
#  be used to update this file.

STARTUP_BOOT_STRAP=com.chelseasystems.cr.appmgr.bootstrap.InitialBoo
tStrap

#  comma delimited class name that need to be executed before any
#  applications are loaded.

BOOT_STRAP=com.chelseasystems.cr.appmgr.bootstrap.ClientLockingBootS
trap,com.chelseasystems.cr.appmgr.bootstrap.ClientServicesBootStrap,
com.chelseasystems.cs.util.StartupBrowserBootStrap,com.chelseasystem
s.cs.swing.GUIBootStrap
```

Logging

This parameter contains logging information related specifically to the SIM client. The retailer can specify the log filename. The system's default uses the directory specified, and separate log files can exist for unique clients. The LOGGING_PAUSE value tells the system how long to wait before writing data to the log file.

For example:

```
#logging

LOGGING_IMPL=com.chelseasystems.cr.logging.LoggingFileServices

#  uncommenting this line will use this as the client log filename
#  rather than the name that includes the timestamp

#LOGGING_FILE_NAME=..\\log\\sim_client.log

LOGGING_LEVEL=4

LOGGING_PAUSE=5000

LOGGING_SYSTEM_OUT=true

LOGGING_SYSTEM_ERR=true
```

Online help-related parameters

The values established in this section are associated with the system's handling of its online help. Under normal operating conditions, the values should *not* have to be changed.

For example:

```
#  Set whether the application will allow the Eu to turn 'Build Help
#  Mode' on ENABLE_BUILD_HELP_MODE=false


#  Sets the base directory to the front page of the help system.
#  The application will attempt to append the user's locale to the
#  end of this value so that the user may see the documentation in
#  his/her locale (if the documentation exists.)  If the application
#  is unable to find the documentation for the user's locale, it
#  will use the base directory as-is (without any locale values
#  appended,) and try to use that.  This value is used in
#  conjunction with the HELP_ENTRY_PAGE value.

#  This value may be:

#  1. a relative path, in which case the path is taken to be
#  relative from the files/prod, or files/training directory.  So a
#  value of "help" would translate to
#  "<rss-dir>files/prod/help-<user-locale>" or
#  "<rss-dir>files/prod/help."

#  2. an absolute file path.  Ex: "y:\\rss-help"

#  3. a fully specified url.  Do not leave the protocol section off,
#  or things will likely not work.  Ex: "http://ourhelpmachine"

HELP_BASE_DIR=OnlineHelp/WebHelp


#  See the comments for HELP_BASE_DIR for details on user locale
#  settings.

#  This value tells the application where then entry page is for the
#  documentation.

HELP_ENTRY_PAGE=iso_im.htm

#  This value holds the executable for displaying help.

#  Available parameters: {0} - the absolute path of the URL which
#  should display the help documentation.

#  (1) - the absolute base directory of the application.

#This may be used to find subsequent files.

#HELP_EXECUTABLE=rundll32 url.dll,FileProtocolHandler {0}

HELP_EXECUTABLE=cmd /c start {0}

#HELP_EXECUTABLE=wscript /nologo {1}OnlineHelp/IEStarter.js {0}
```

List of resource bundles

Resource bundles are Java files related to the internationalization process. This entry specs the system's default resource bundles. Language-specific versions are handled automatically based on the configured user's locale. These values should *not* have to be changed even when the system is offered in different languages.

For example:

```
# Comma-delimited list of classes of resource bundles

MESSAGE_BUNDLE=com.chelseasystems.cs.util.MessageBundle,com.chelseas
ystems.cs.util.LabelResourceBundle,com.chelseasystems.cs.util.Button
ResourceBundle,com.chelseasystems.cs.util.RSSRuleMessageBundle,com.c
helseasystems.cs.util.RendererResourceBundle,com.chelseasystems.cs.u
til.EMailTextResourceBundle,com.chelseasystems.cs.util.RSSRibResourc
eBundle
```

Look and feel of the client

These parameters allow the retailer to customize the appearance of the application on the PC. However, Retek does *not* recommend that these values be changed.

For example:

```
#  Attempt to use one of the following Look And Feels from LF1 to
#  LFn.

LOOK_FEEL_KEY=LF1,LF2,LF3,LF4,LF5

LF1=com.chelseasystems.cr.swing.plaf.metal.ConfigurableMetalLookAndF
eel

LF2=com.apple.mrj.swing.MacLookAndFeel

LF3=com.sun.java.swing.plaf.windows.WindowsLookAndFeel

LF4=javax.swing.plaf.metal.MetalLookAndFeel

LF5=com.sun.java.swing.plaf.motif.MotifLookAndFeel


# The direction the toolbar and appbar will show up

#     0 - Bottom Right

#     1 - Bottom Left

#     2 - Upper Right

#     3 - Upper Left

USER_PREFERENCES.ANCHOR=0

# Show a splash

SHOW_SPLASH=FALSE
```

```
#  The milliseconds delay to wait before enabling a button on a
#  newly displayed applet

# Suggested value: 200 (two tenths of a second)

BUTTON_DELAY=0

#  whether the button text should be displayed with HTML.  Useful
#  for word-wrapping.

FORMAT_BUTTONS_WITH_HTML=false
```

Port to lock the client application

SIM does *not* use this value except in a peripheral way. The retailer could change the port value if the specified port is used for another purpose on the retailer's system.

For example:

```
#  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=51802
```

Client services loaded by bootstraps

These parameters are associated with the initial system processing of client services. Under normal operating conditions, these parameters should *not* have to be changed.

For example:

```
#Client Services loaded by bootstraps

SERVICES_LIST=CONFIG_SRVC,EMPLOYEE_SRVC,AUTH_SRVC,LOCALITEM_SRVC,MDS
EHIERARCHY_SRVC,MINISTORE_SRVC,PRINTING_SRVC,REGISTER_SRVC,SOURCE_SR
VC,STOCKCOUNT_SRVC,STOCKCOUNTEVENT_SRVC,STOCKEVENT_SRVC,STORE_SRVC,T
XN_NUMBER_SRVC,TXN_POSTER_SRVC,ACTIVITYLOCKING_SRVC,PROCESSMEASUREAU
DIT_SRVC,SEQUENCING_SRVC,REPLENISHMENT_SRVC,PRODUCTGROUP_SRVC,ADHOCC
OUNTADMIN_SRVC
```

Logging information

For information about logging related to the parsing processing of the PosmodDnldFileParser batch process and the DexnexFileParser batch process, see “Chapter 7 – Java batch processes”.

Default location of client and server log files

Unix

Log files related to the client are located in the following directory:

```
clientUnix\retek\sim\files\prod\log
```

Log files related to the server are located in the following directory:

```
serverUnix\retek\sim\files\prod\log
```

Windows

Log files related to the client are located in the following directory:

```
clientWindows\retek\sim\prod\log
```

Log files related to the server are located in the following directory:

```
serverWindows\retek\sim\files\prod\log
```

Logging levels established in configuration files (.cfg)

Logging levels can be established configuration files (for example, stockcount.cfg, store.cfg, and activitylocking.cfg, and so on). This level of logging can be helpful when troubleshooting specific parts of the application. For example, if the application is experiencing issues in a specific area, the logger can set to a higher degree of granularity. For example, the stockcount.cfg file contains the following entry:

```
# Logging
LOGGING_IMPL=com.chelseasystems.cr.logging.LoggingFileServices
LOGGING_FILE_NAME=../log/sim_stockcount.log
LOGGING_LEVEL=4
LOGGING_PAUSE=5000
LOGGING_SYSTEM_OUT=true
LOGGING_SYSTEM_ERR=true
```

Exception handling

The primary types of exceptions within the SIM system include the following:

- com.chelseasystems.cr.rules.BusinessRuleException
  This exception indicates that a business rule has been violated.
- com.chelseasystems.cr.appmgr.ApplicationException
  This exception is used to wrap lower system-level exceptions that occur throughout the framework. This class should be used to specify what the application needs to handle this exception with business logic, allowing lower level exceptions that occur to fall through and be handled differently.
- com.chelseasystems.cs.dataaccess.DAOException
  A DAOException is an exception thrown by the DAO. When some code has caught an exception in a DAO, it should throw a DAOException upward as a result. This class allows the original exception to continue to be accessible.
- com.chelseasystems.cr.activitylocking.ActivityLockingException
  This exception class extends BusinessRuleException and provides an API to provide the String identifier of the current lock holder.

Java Virtual Machine (JVM) options

Any JVM modifier (that is, any that can be specified in the ‘Java’ program) can be specified in the RSSMainContainer.xml file.

For example, if out of memory issues occur because of the amount of data involved during runtime, the JVM setting can be adjusted in the section of the RSSMainContainer.xml file shown below. Note that in the example below the 128m stands for 128 megabytes, the value that would need to be increased.

For example:

```
<jvmLineArgs length="2">
            <java.lang.String>-Xms4m</java.lang.String>
            <java.lang.String>-Xmx128m</java.lang.String>
        </jvmLineArgs>
```

Default number of threads setting for HP-UX retailers only

The default number of threads per process is set to a small number (64 for example). This number does *not* currently allow the system to function properly for HP-UX retailers, and the number must be changed to a much higher number (5000 for example).

Chapter 3 – Technical architecture

This chapter describes the overall software architecture for SIM, offering a high-level discussion of the general structure of the system, including the various layers of Java code. This information is valuable in the following scenarios, among others:

- When the retailer wishes to take advantage of SIM's extensible capabilities and write its own code to fit into the SIM system.
- When the retailer wishes to implement the system for various databases (Oracle, DB2, and so on).

For those who are less familiar with Java terminology, a description of SIM-related Java terms and standards is provided for your reference at the end of this chapter.

Overview

SIM and Integrated Store Operations (ISO)

ISO is a group of in-store operations applications that will use and share common business logic, configuration utilities and technology infrastructure. The applications are proven to simplify store operations by improving customer service and decreasing costs. SIM is an application on the ISO platform.

Advantages of the architecture

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

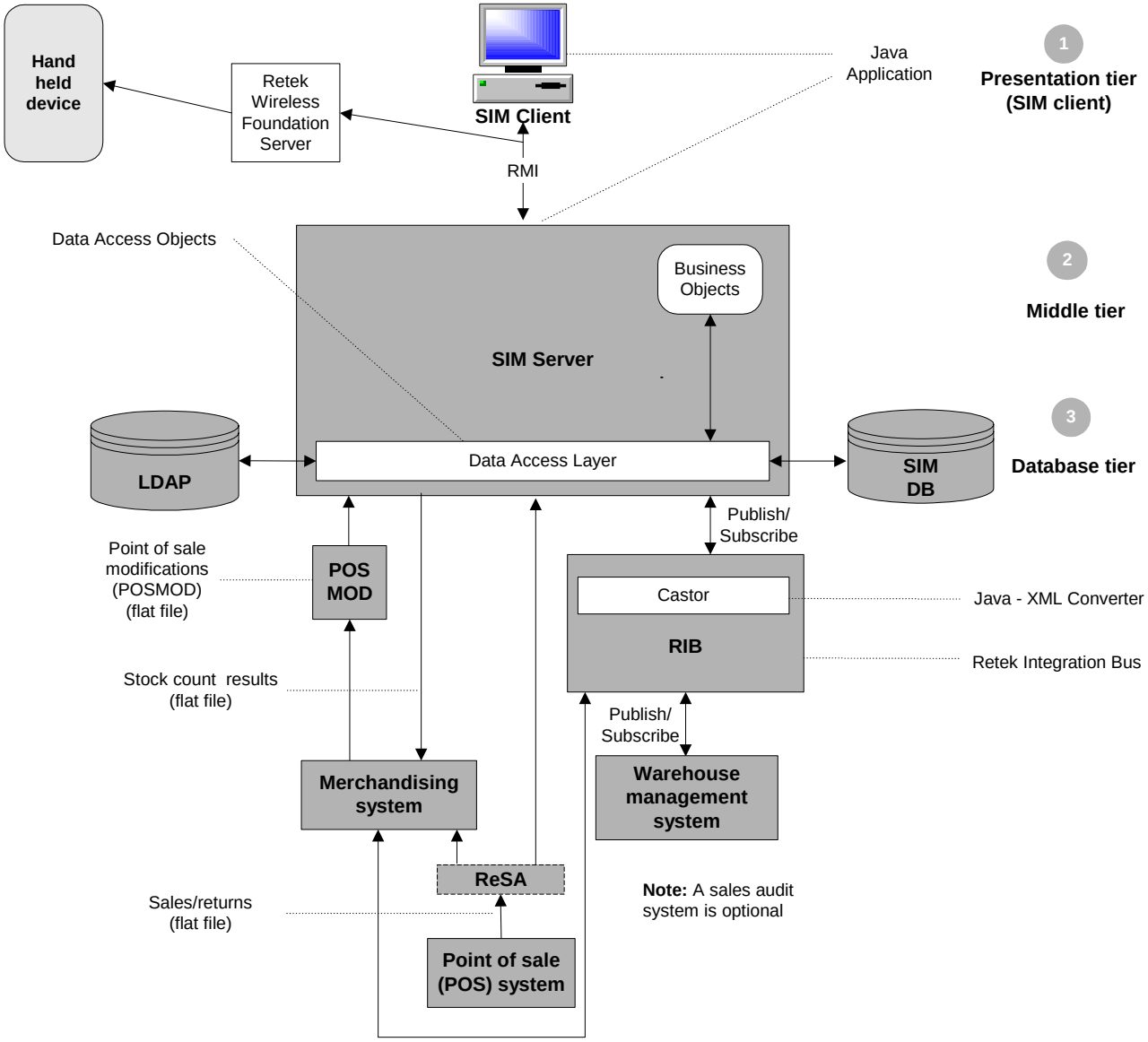
The n-tier architecture of SIM allows for the encapsulation of business logic, shielding the client from the complexity of the back-end system. Any given tier need not be concerned with the internal functional tasks of any other tier.

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

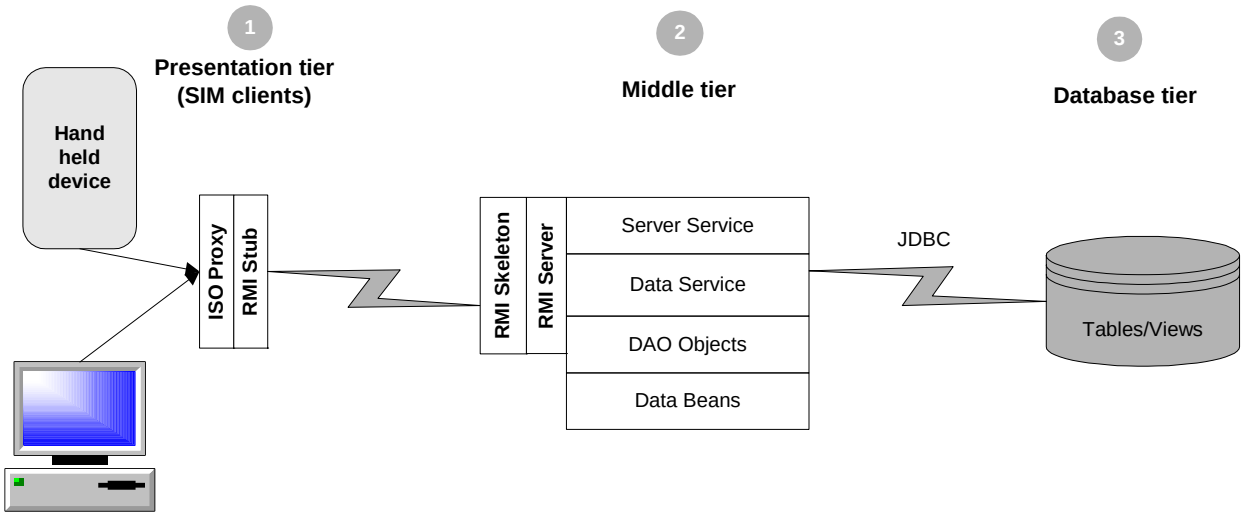
- Scalability: Hardware and software can be added to meet retailer requirements for each of the tiers.
- Maintainability: The separation of presentation, business logic, and data makes the software cleaner, more maintainable, and easier to modify.
- Platform independence: The code is written once but 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. The diagrams below illustrate the major pieces of the typical three-tiered SIM implementation. Descriptions follow both diagrams for the numbered items.



SIM's technical architecture within the enterprise



SIM's technical architecture

Presentation tier

1 SIM can be deployed on a wide variety of clients, including a desktop computer, a hand-held wireless device, and so on. The GUI is responsible for presenting data to the user and for receiving data directly from the user through the 'front end'. The presentation tier only interacts with the middle tier (as opposed to the database tier). To optimize performance, the SIM 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 graphical user interfaces (GUIs) in Java applications.

The handheld communication infrastructure piece, known as the Retek Wireless Foundation Server, enables the handheld devices to communicate successfully. The handheld devices 'talk' to the Retek Wireless Foundation Server, which in turn acts as a client to SIM's middle tier.

Middle tier

2 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 beans (that is, Java classes that have one or more attributes and corresponding set / get methods) that represent a functional entity. In other words, business objects can be thought of as data containers, which by themselves have almost no business functionality.

Although the PC client and the handheld client use the middle tier's functionality differently, the middle tier is the same for both clients. For example, the handheld device, used 'on the fly', performs frequent commits to the database, while the PC performs more infrequent commits. The application is flexible in that it accommodates the different styles of client-driven processing.

The middle tier is designed to function as a 'stateless' object, meaning it receives whatever instruction it needs to access the database from the client. Further, SIM has failover abilities because, with only a very few exceptions, none of the data passed to any container from the database is ever cached and stored in the middle tier between client interactions. In other words, if a specific middle tier server fails, processing can roll over to another SIM server for continued processing.

In addition, 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, the platform does not need separate configurations for each application server. The result of this feature is that SIM retailers are never aware that additional application servers have been added to help with the workload. SIM application servers contain multiple containers, each of which is related to a unique Java Virtual Machine (JVM). Each container can have several instances of a component, and each component corresponds to a specific SIM service. Introducing multiple instances of a container or service 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.
- The DAO objects are abstract allowing for database independence.
- Data services 'talk' to the relevant data beans and build the necessary objects.
- Data beans that create the necessary SQL are used to retrieve data from the database. Note that a one-to-one relationship exists between the data beans and the tables/views contained in the database.

Data access objects (DAO)

DAOs are classes that abstract the actual persistence mechanism that is being used to persist business objects. The DAO objects provides the mechanism that allows SIM to be associated to a different persistence engine. Ideally, in those cases, only the DAO would need to be modified due to the change. The remainder of SIM would continue to operate unchanged.

Java database connectivity (JDBC)

The middle-tier talks with the database via the industry standard Java database connectivity (JDBC) protocol. In order for the SIM client to retrieve the desired data from the database, the 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 (API)s that offer a database-independent means of extracting and/or inserting data to or from SIM. To perform those insertions and extractions, SQL code also resides in this tier facilitating create, read, update, and delete actions. For a definition of JDBC, see the "SIM-related Java terms and standards" section in this chapter.

Database tier

The SIM data model includes some tables and columns that are SIM-specific and some that derive their names from the Association for Retail Technology Standards (ARTS) data model. Note, though, that SIM uses but does not fully conform to the ARTS standard.

3 The database tier is the application's storage platform, containing the physical data (user and system) used throughout the application. The database houses the tables, views and records which are retrieved by the container and then passed to the client. Note that the database does not generate any additional PL/SQL beyond that which is passed to it via JDBC. This tier is only intended to deal with the storage and retrieval of information and is not involved in the manipulation or in the delivery of the data. This tier responds to queries; it does not initiate them.

   SIM supports multiple database types and persistence mechanisms (for example, Oracle, DB2, and so on).

Distributed topology

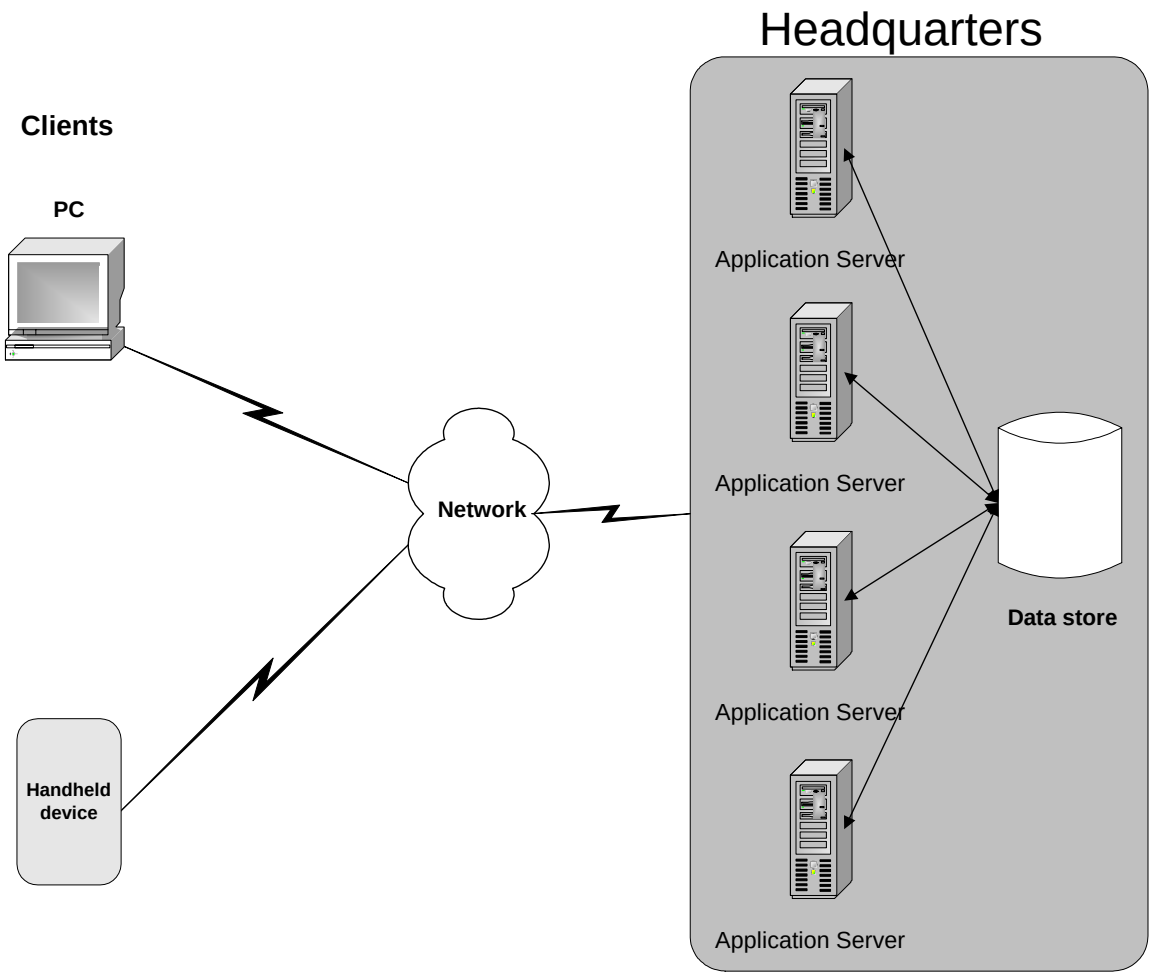
One of SIM's most significant advantages is its flexible distributed topology. SIM offers complete location transparency because the location of data and/or services is based upon the retailer's business requirements, not upon technical limitations. SIM's client server connection utilizes Remote Method Invocation (RMI). For SIM, the use of RMI means that the application can take advantage of distributed objects; that is, the server can be geographically distributed, residing at a central location. 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 RMI stub resides on the client side of the client/server relationship. The RMI stub contains no business logic but contains only enough code to effectively 'pass through' data. As far as the clients are concerned, all the processing occurs locally because they interact only with the stub, which is local. The GUI uses helper classes to 'talk' with the stub.

For example, if the stub is asked through the client to perform the method 'update shipment', the RMI stub appears to have that capability, though in reality it only behaves as a passage to the RMI skeleton, which resides on the server side. The skeleton 'talks' with the server to complete the business-logic driven processing. The server performs the actual 'update shipment' business logic.

Connectivity between the SIM client and the middle tier is achieved via a Remote Naming Service (RNS), which provides the SIM client with the necessary IP address and port. The remote naming service (RNS), or middleware tier of the SIM platform, essentially serves as a communication bridge between the clients and the application server.

The following diagram illustrates SIM's deployment.



SIM's deployment

Packages containing key services and business objects

The following packages, viewable in the Javadoc, contain classes that are integral to the system's business processing. These classes contain both business logic methods and look up methods and are located in the middle tier.

- com.chelseasystems.cr.group
- com.chelseasystems.cr.localitem
- com.chelseasystems.cr.mdsehierarchy
- com.chelseasystems.cr.print
- com.chelseasystems.cr.replenishment
- com.chelseasystems.cr.rules
- com.chelseasystems.cr.sequencing
- com.chelseasystems.cr.stock.activity
- com.chelseasystems.cr.stock.adhoc
- com.chelseasystems.cr.stock.count
- com.chelseasystems.cr.stock.countgroup
- com.chelseasystems.cr.stockevent
- com.chelseasystems.cr.store
- com.chelseasystems.cr.swing
- com.chelseasystems.cr.pricechange
- com.chelseasystems.cr.storeorder
- com.chelseasystems.cs.activitylocking
- com.chelseasystems.cs.group
- com.chelseasystems.cs.localitem
- com.chelseasystems.cs.mdsehierarchy
- com.chelseasystems.cs.print
- com.chelseasystems.cs.replenishment
- com.chelseasystems.cs.rules
- com.chelseasystems.cs.sequencing
- com.chelseasystems.cs.source
- com.chelseasystems.cs.stock.count
- com.chelseasystems.cs.stock.countgroup
- com.chelseasystems.cs.stock.event
- com.chelseasystems.cs.stockevent
- com.chelseasystems.cs.store
- com.chelseasystems.cs.swing
- com.chelseasystems.cs.pricechange
- com.chelseasystems.cs.storeorder

A word about 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.

Technical support services

Technical services hold the application together by providing common services to the application, services that are not necessarily driven by business requirements. In order to increase the maintainability of the code, a number of base technical services are provided in SIM. They are described below.

Transaction service

SIM’s transaction service layer is responsible for providing two-phase commit transaction support. This processing ensures that multiple persistence layer changes are either all committed or all rolled back.

Logging service

This service provides the system with a standard method for logging information to a flat text file.

Internationalization service

This service uses Java classes with a .java extension to provide configurability for on-screen messages (such as on screen labels or error messages). To change the language for the SIM GUI screens, the retailer can edit these .java files and recompile them without impacting the business functionality of the application. Note that although this service supports any number of languages, the screen flow remains left to right, top to bottom. SIM uses standard Java resource bundle support.

Note that the locale of a user is specified in his or her LDAP configuration. That is, the preferred language of a user is associated with his or her user ID and *not* with the machine itself.

Security service

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

Security and Light Directory Access Protocol (LDAP)

The LDAP standard defines a network protocol for accessing information in a directory.

LDAP is the persistence mechanism that SIM uses to store authentication and authorization information. SIM reads standard user information from an LDAP server. The LDAP browser is the means by which users are set up in the system.

The security framework within SIM encompasses the following two areas of concern:

- Authentication: Determines whether the user has permission to use the system.
- Authorization. Determines which privileges/roles a user has and which stores or locations the user can perform the functions in.

An LDAP server can have multiple schemas (one might point to SIM, another to POS, and so on). The configuration file determines how schemas are associated to LDAP servers. See Chapter 2, "Backend system configuration".

The LDAP setup takes place through the LDAP browser. The retailer establishes users who are assigned to roles (which govern the activity privileges) and associated to stores. The roles are reflected within the GUI of the SIM application. Note that stores set up with LDAP must exist in the applicable store table of the SIM database.

Users must be assigned to a valid store and given valid roles. The system validates an 'entered' user ID and password with what has been established. Once the system accepts the user and checks the user's roles, the system limits the user's view of the application according to his or her role. Note that a 'superuser' has all privileges.

Additional LDAP resources

- http://www.openldap.org/
  This site contains the OpenLDAP main page. This site contains introduction, downloads, and documentation.
- http://www.iit.edu/~gawojar/ldap/
  This site is an LDAP browser site.

SIM-related Java terms and standards

SIM is deployed using the technologies, methods, versions and/or design patterns defined in this section.

Data access object (DAO)

This design pattern isolates data access and persistence logic. The rest of the component can thus ignore the persistence details (the database type or version, for example).

Instantiate

In the context of SIM, to 'instantiate' means to create a new instance of a class.

Java Development Kit (JDK), version 1.3.1

Standard Java development tools from Sun Microsystems.

Java Messaging Service (JMS) topic

A JMS topic is part of Retek's message-oriented middleware. SIM uses a JMS view of the RIB. The topic can be thought of as broadcasting a message from the RIB. SIM (and potentially other subscribers) can subscribe to that broadcast.

JDBC

JDBC is a means for Java-architected applications such as SIM to execute SQL statements against an SQL-compliant database, such as Oracle. Part of Sun's J2EE specification, most database vendors implement this specification.

Naming conventions in Java

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

Persistence

The protocol for transferring the state of a business object between variables and an underlying database.

Persistent connections

The state of connection between an application and the database.

Remote interface

The client side interface to a service. This interface defines the server-side methods available in the client tier.

Remote Method Invocation (RMI)

RMI offers a simple and direct model for distributed computation with Java objects. Remote method invocation (RMI) is the action of invoking a method of a remote interface on a remote object. Most importantly, a method invocation on a remote object has the same syntax as a method invocation on a local object.

Skeleton

The skeleton understands how to communicate with the stub across the RMI link. The skeleton performs all of the following:

- 'Talks' with the stub.
- Reads the parameters for the method call from the link.
- Makes the call to the remote service implementation object, accepts the return value, and then writes the return value back to the stub.

Stub

The stub contains information that allows it to connect to a remote object, which contains the implementation of the methods. The stub implements the same set of remote interfaces as the remote object's class.

Chapter 4 – SIM and the Retek Integration Bus (RIB)

SIM and the RIB overview

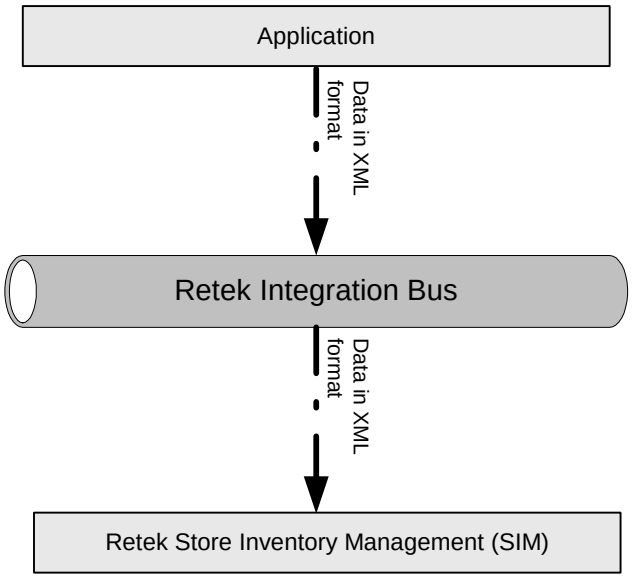
The flow diagrams and explanations in this chapter provide a brief overview of publication and subscription processing.

Retek utilizes a publish and subscribe (pub/sub) messaging paradigm with some guarantee of delivery for a message. In a pub/sub messaging system, an adapter publishes a message to the integration bus that is then forwarded to one or more subscribers. The publishing adapter does not know, nor care, how many subscribers are waiting for the message, what types of adapters the subscribers are, what the subscribers' current states are (running/down), or where the subscribers are located. Delivering the message to all subscribing adapters is the responsibility of the integration bus.

See the latest Retek Integration Guide and other RIB-related documentation for additional information.

The XML message format

As shown by the diagram below, the messages to which SIM subscribes are in an XML format and have their data structure defined by document type definitions (DTDs) or XML schema documents.



Data across the RIB in XML format

Message subscription processing

Message subscription processing diagram

From a high-level perspective, all of SIM's message subscription processes primarily follow the same pattern. The flow diagram and the numbered explanations below provide a brief overview to this subscription process. Note that the explanation utilizes, as its example, SIM's subscription to a flat Diff message consisting of a Diff record.

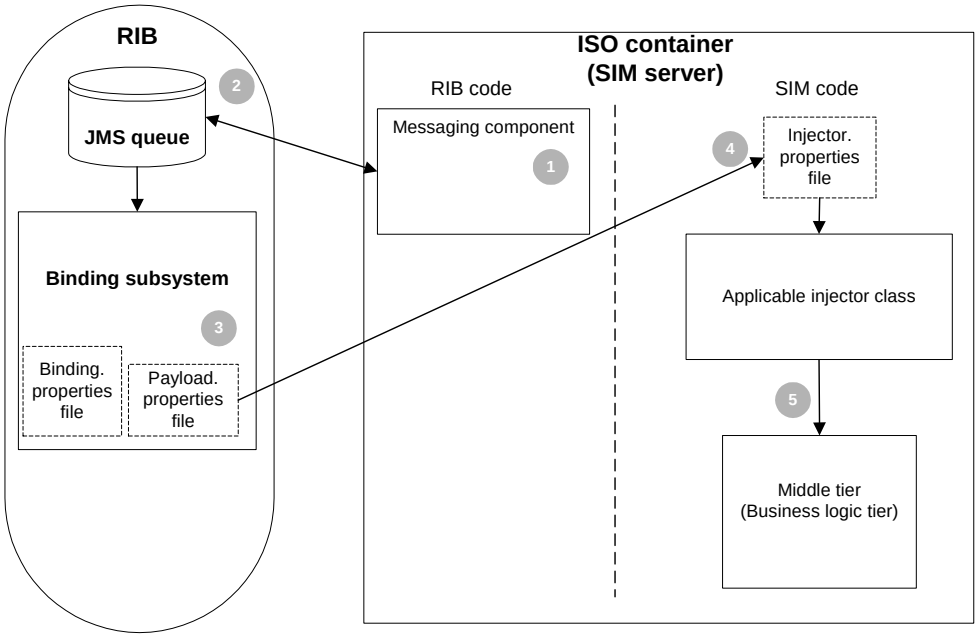


Diagram of SIM's message subscription process

Message subscription processing description

1 The messaging component is derived from SIM Java classes and is deployed inside the ISO container (the SIM server). However, the code for a messaging component is 'owned' by the RIB.

For every message family and type, the messaging component code is the same. When the message component is deployed, it amounts merely to entries and configuration files. The file, RIBContainer.xml, determines what runs in the container. The RIBContainer.xml file includes all instances of information that includes the messaging component and the configuration file name. A summary of this information is provided later in this chapter.

2 On the RIB, a message is delivered to a JMS topic to which SIM subscribes. Once the JMS topic (for example, EtDiffsFromRMS) learns that a (diff-related) message in XML format exists, the Java-based message component (DiffsMessagingComponent, for example) is notified to get the message. The RIB Java Messaging Service (JMS) continues the process. For a definition of JMS topic, see the section, 'SIM-related Java terms and standards' in Chapter 3, "Technical architecture".

3 Referencing both the binding.properties file and the payload.properties file, the Binding uses an XML parsing utility to parse the XML and transform the RIB message's three parts (the family, the type, and the payload) into a Java payload (object) that SIM's middle tier can interpret. The term 'Binding' is used to identify a message processing subsystem that is used for Retek's ISO and J2EE environments. This subsystem consists of a set of classes used to process or create RIB messages. The Binding is responsible for translating the RIB Message payload XML string into a 'payload' object and to execute application specific code to process the payload. In other words, the Binding turns the XML message into a Java object.

Note: Under ordinary runtime conditions, the integration-related properties files and configuration files described above do not have to be modified.

4 Based on the message family and the message type, an injector.properties file within SIM knows which subscribing java class (called 'injectors' within the code) to instantiate. The injector class implements an application messenger injector interface. Note that one injector can handle multiple .dtd messages. For a definition of instantiate, see the section, 'SIM-related Java terms and standards' in Chapter 3, "Technical architecture".

5 The subscriber 'injects' the data into the application's middle tier, which is configured to act upon and/or validate the data. When SIM receives data from the merchandising system via the RIB, SIM behaves as if the RIB were a client to the system. The RIB becomes a system interface (SI) to SIM (in other words, the RIB becomes an interface without a GUI). SIM processes the RIB data in the same manner as it would data entered by a person through the graphical user interface (GUI).

Subscription-related RIB error messages

For any given message to which SIM subscribes, the injector handles a message from its entry point into the SIM system all the way to the database. If a message fails in the injector-related process and an exception is thrown, a number of messages can be returned, including the following:

- CREATE_FAILED
- UNMARSHALLING_ERROR
- COMMAND_FACTORY_CANNOT_INSTANTIATE_PAYLOAD
- MODIFY_FAILED

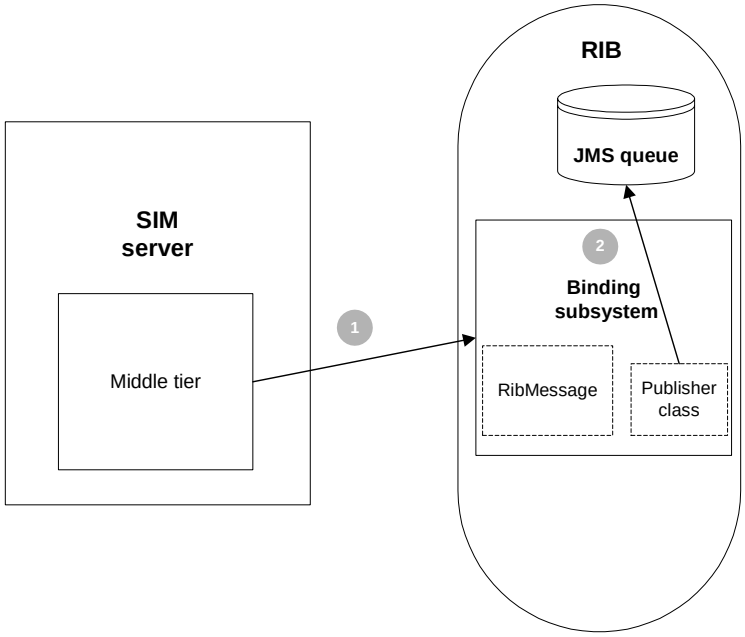
RIB error hospital

The RIB error hospital is a set of Java classes and database tables located within the SIM server but 'owned' by the RIB. The RIB error hospital is designed to segregate and trigger re-processing for messages that had some error with their initial processing. The intent is to provide a means to halt processing for messages that cause errors while allowing continued processing for the 'good' messages. The RIB hospital references tables within SIM (for example, RIB_MESSAGE, RIB_MESSAGE_FAILURE, RIB_MESSAGE_ROUTING_INFO). For more information about the RIB hospital, see the latest RIB Technical Architecture Guide, RIB Operations Guide, or RIB Hospital Administration online help.

Message publication processing

From a high-level perspective, all of SIM's message publication processes primarily follow the same pattern. The flow diagram and the numbered explanations below provide a brief overview to this publication process.

Message publication processing diagram



SIM message publication processing

Message publication processing description

1 An event within SIM (that is, an insert, update, or delete) leads its middle tier to populate a payload object and to call the Binding subsystem.

2 The Binding subsystem converts the payload object into an XML string. The object on the Binding subsystem is put into a RIB envelope called RibMessage. The data within RibMessage eventually becomes a message on the RIB. A Publisher class in the Binding subsystem is called to write the data to the RIB's JMS queue. On a (configurable) regular basis, the RIB engages in polling the JMS queue, searching for the existence of a message. A publishable message that appears on the JMS queue is processed.

Table summary of messaging component .cfg files

The following table lists the messaging component (subscriber) .cfg files. For more information, see the latest RIB Technical Architecture Guide or RIB Operations Guide.

Messaging component configuration file
asninmessagingcomponent.cfg
asnoutmessagingcomponent.cfg
diffsmessagingcomponent.cfg
itemlocmessagingcomponent.cfg
itemsmessagingcomponent.cfg
itemzoneprcmessagingcomponent.cfg
ordermessagingcomponent.cfg
prcchgconfmessagingcomponent.cfg
prczonegrpmessagingcomponent.cfg
rtvreqmessagingcomponent.cfg
seedmessagingcomponent.cfg
stockordermessagingcomponent.cfg
storesmessagingcomponent.cfg
vendormessagingcomponent.cfg
whmessagingcomponent.cfg

Table summary of publisher files

The table below lists SIM's publisher .cfg files. For more information, see the latest RIB Technical Architecture Guide or RIB Operations Guide.

Publisher files
asnoutpublisher.cfg
dsdreceiptpublisher.cfg
invadjustpublisher.cfg
invreqpublisher.cfg
prcchgreqpublisher.cfg
receivingpublisher.cfg
rtvpublisher.cfg
sostatuspublisher.cfg
stkcountschpublisher.cfg

Subscribers mapping table

The following table lists the message family and message type name, the document type definition (DTD) that describes the XML message, and the subscribing classes that facilitate the data's entry into the application's middle tier. These classes are described in the code as 'injectors'. For additional information, see the latest Retek Integration Guide and other RIB documentation.

Family	Type	DTD/Payload	Subscribing class ('injector')
ASNIN	ASNOUTCRE	ASNInDesc	ASNInCreateInjector
ASNIN	ASNINDEL	ASNInRef	ASNInRemoveInjector
ASNIN	ASNINMOD	ASNInDesc	ASNInModifyInjector
ASNOUT	ASNOUTCRE	ASNOutDesc	ASNOutCreateInjector
DIFFS	DIFFCRE	DiffDesc	DifferentiatorCreateInjector
DIFFS	DIFFDEL	DiffRef	DifferentiatorRemoveInjector
DIFFS	DIFFMOD	DiffDesc	DifferentiatorModifyInjector
ITEMS	ITEMBOMCRE	ItemBOMDesc	ItemBOMCreateInjector
ITEMS	ITEMBOMDEL	ItemBOMRef	ItemBOMRemoveInjector
ITEMS	ITEMBOMMOD	ItemBOMDesc	ItemBOMModifyInjector
ITEMS	ITEMCRE	ItemDesc	ItemCreateInjector
ITEMS	ITEMDEL	ItemRef	ItemRemoveInjector

Family	Type	DTD/Payload	Subscribing class ('injector')
ITEMS	ITEMHDRMOD	ItemHdrDesc	ItemModifyInjector
ITEMS	ITEMSUPCRE	ItemSupCtyDesc	ItemSupCreateInjector
ITEMS	ITEMSUPCTYCRE	ItemSupCtyRef	ItemSupCtyCreateInjector
ITEMS	ITEMSUPCTYDEL	ItemSupCtyRef	ItemSupCtyRemoveInjector
ITEMS	ITEMSUPCTYMOD	ItemSupCtyDesc	ItemSupCtyModifyInjector
ITEMS	ITEMSUPDEL	ItemSupRef	ItemSupRemoveInjector
ITEMS	ITEMSUPMOD	ItemSupDesc	ItemSupModifyInjector
ITEMS	ITEMUPCCRE	ItemUPCDesc	ItemUPCCreateInjector
ITEMS	ITEMUPCDEL	ItemUPCRef	ItemUPCRemoveInjector
ITEMS	ITEMUPCMOD	ItemUPCDesc	ItemUPCModifyInjector
ITEMZONEPRC	ITEMZONEPRCCRE	ItemZonePrcDesc	ItemZonePrcCreateInjector
ITEMZONEPRC	ITEMZONEPRCDEL	ItemZonePrcRef	ItemZonePrcRemoveInjector
ITEMZONEPRC	ITEMZONEPRCMOD	ItemZonePrcDesc	ItemZonePrcModifyInjector
ORDER	POCRE	PODesc	PurchaseOrderCreateInjector
ORDER	PODEL	PORef	PurchaseOrderRemoveInjector
ORDER	PODTLCRE	PODesc	PurchaseOrderDetailCreateInjector
ORDER	PODTLDEL	PORef	PurchaseOrderDetailRemoveInjector
ORDER	PODTLMOD	PODesc	PurchaseOrderDetailModifyInjector
ORDER	POHDRMOD	PODesc	PurchaseOrderModifyInjector
PRCCHGCONF	PRCCHGCONFCRE	PrcChgConfDesc	PrcChgConfCreateInjector
PRCZONEGRP	PRCZONEGRPCRE	PrcZoneGrpDesc	PrcZoneGrpCreateInjector
PRCZONEGRP	PRCZONEGRPDEL	PrcZoneGrpRef	PrcZoneGrpRemoveInjector
PRCZONEGRP	PRCZONEGRPMOD	PrcZoneGrpDesc	PrcZoneGrpModifyInjector
RTVREQ	RTVREQCRE	RTVReqDesc	RTVReqCreateInjector

Family	Type	DTD/Payload	Subscribing class ('injector')
RTVREQ	RTVREQMOD	RTVReqDesc	RTVReqModifyInjector
RTVREQ	RTVREQDEL	RTVReqRef	RTVReqRemoveInjector
RTVREQ	RTVREQDTLCRE	RTVReqDesc	RTVReqDetailCreateInjector
RTVREQ	RTVREQDTLDEL	RTVReqRef	RTVReqDetailRemoveInjector
RTVREQ	RTVREQDTLMOD	RTVReqDesc	RTVReqDetailModifyInjector
SEEDDATA	DIFFTYPECRE	DiffTypeDesc	DifferentiatorTypeCreateInjector
SEEDDATA	DIFFTYPEDEL	DiffTypeRef	DifferentiatorTypeRemoveInjector
SEEDDATA	DIFFTYPEMOD	DiffTypeDesc	DifferentiatorTypeModifyInjector
STOCKORDER	SOCRE	SODesc	StockOrderCreateInjector
STOCKORDER	SODTLCRE	SODesc	StockOrderCreateInjector
STOCKORDER	SODTLDEL	SORef	StockOrderRemoveInjector
STOCKORDER	SODTLMOD	SODesc	StockOrderModifyInjector
STOCKORDER	SOHDRDEL	SORef	StockOrderRemoveInjector
STOCKORDER	SOHDRMOD	SODesc	StockOrderModifyInjector
STORES	STORECRE	StoresDesc	StoreCreateInjector
STORES	STOREDEL	StoresRef	StoreRemoveInjector
STORES	STOREMOD	StoresDesc	StoreModifyInjector
VENDOR	VENDORADDRCRE	VendorAddrDesc	SupplierAddrCreateInjector
VENDOR	VENDORADDRDEL	VendorAddrRef	SupplierAddrRemoveInjector
VENDOR	VENDORADDRMOD	VendorAddrDesc	SupplierAddrModifyInjector
VENDOR	VENDORCRE	VendorDesc	SupplierCreateInjector
VENDOR	VENDORDEL	VendorRef	SupplierRemoveInjector
VENDOR	VENDORHDRMOD	VendorHdrDesc	SupplierModifyInjector
WH	WHCRE	WHDesc	WareHouseCreateInjector
WH	WHDEL	WHRef	WareHouseRemoveInjector
WH	WHMOD	WHDesc	WareHouseModifyInjector

Publishers mapping table

This table illustrates the relationship among the message family, message type and the DTD/payload object that the application creates. For additional information, see the latest Retek Integration Guide and other RIB documentation.

Family	Type	DTD/Payload
ASNOUT	ASNOUTCRE	ASNOutDesc
DSDRECEIPT	DSDRECEIPTCRE	DSDReceiptDesc
INVADJUST	INVADJUSTCRE	InvAdjustDesc
INVREQ	INVREQCRE	InvReqDesc
PRCCHGREQ	PRCCHGREQCRE	PrcChgReqDesc
RECEIVING	RECEIPTCRE	ReceiptDesc
RECEIVING	RECEIPTMOD	ReceiptDesc
RTV	RTVCRE	RTVDesc
SOSTATUS	SOSTATUSCRE	SOStatusDesc
STKCOUNTSCH	STKCOUNTSCHCRE	StkCountSchDesc
STKCOUNTSCH	STKCOUNTSCHDEL	StkCountSchRef
STKCOUNTSCH	STKCOUNTSCHMOD	StkCountSchDesc

Functional descriptions of messages

The table below briefly describes the functional role that messages play with regard to SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
ASN in	Subscription	RDM, Vendor (external)	These messages contain inbound shipment notifications from both vendors (PO shipments) and warehouses (transfer and allocation shipments).
ASN out	Publication	RMS, RDM	These messages are used by SIM to communicate store-to-warehouse transfers (returns to warehouse) to both RMS and RDM. These messages are also used to communicate store-to-store transfers to RMS.
Diff IDs	Subscription	RMS	These messages are used to communicate differentiator IDs from RMS to SIM.
DSD receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of a supplier delivery for which no RMS purchase order had previously existed.
Items	Subscription	RMS	These are messages communicated by RMS that contain all approved items records, including header information, item/supplier, and item/supp/country details, and item/ticket information.

Functional area	Subscription/ publication	Integration to Products	Description
Item/location	Subscription	RMS	These are messages communicated by RMS that contain item/location data used for ranging of items at locations and communicating select item/location level parameters used in store order and item request.
Item zone price	Subscription	RMS	These are messages communicated by RMS that contain all item/zone/price records. This data is used by SIM so that a store knows an item's retail price before the item becomes ranged at a location.
Inventory adjustments	Publication	RMS	These messages are used by SIM to communicate inventory adjustments. RMS uses these messages to adjust inventory accordingly.
Inventory request	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item. RMS uses this data to fulfill the requested inventory through either auto-replenishment or by creating a one-off purchase order/transfer.
Purchase orders	Subscription	RMS	These messages contain approved, direct to store purchase orders. SIM uses these to receive direct deliveries against.

Functional area	Subscription/ publication	Integration to Products	Description
Price change confirm	Subscription	RMS	These messages contain a confirmation for a price change request. SIM uses this type of message to update the status of a price change created in SIM to reflect the price change's status created by RMS.
Price change request	Publication	RMS	These messages contain a request for a price change. RMS uses this message to create a price change and then to perform conflict checking against the price change, and return a confirmation to SIM including the status of the price change.
Price zone groups	Subscription	RMS	These messages are communicated by RMS containing the price zone group/zone/store hierarchies. This data is used in conjunction with the item/zone/price records to determine an item's retail price at a location.
Receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of an RMS purchase order, a transfer, or an allocation.
Receiver unit adjustments	Publication	RMS	These messages are used by SIM to communicate any adjustments to the receipts of purchase orders, transfers, and allocations. These messages are part of the RECEIVING message family (receiving unit adjustments only use the RECEIPTMOD message type).

Functional area	Subscription/ publication	Integration to Products	Description
Return to vendor	Publication	RMS	These messages are used by SIM to communicate the shipment of a return to vendor from the store.
RTV request	Subscription	RMS	These are messages communicated by RMS that contain a request to return inventory to a vendor.
Seed data	Subscription	RMS	These messages communicated by RMS contain differentiator type values.
Stock count schedules	Publication	RMS	These messages are used by SIM to communicate unit and value stock count schedules to RMS. RMS uses this schedule to take an inventory snapshot of the date of a scheduled count.
Stock order status	Publication	RMS	These messages are used by SIM to communicate the cancellation of any requested transfer quantities. For example, the merchandising system can create a transfer request for 90 units from a store. If the sending store only ships 75, a cancellation message is sent for the remaining 15 requested items.
Stores	Subscription	RMS	These are messages communicated by RMS that contain stores set up in the system (RMS).
Store ordering	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item. This functionality represents an extension of item request functionality and uses the same e*ways.

Functional area	Subscription/ publication	Integration to Products	Description
Transfer request	Subscription	RMS	These messages are communicated by RMS and contain a request to transfer inventory out of a store. Upon shipment of the requested transfer, SIM uses the ASN Out message to communicate what was actually shipped. In addition, SIM uses the stock order status message to cancel any requested quantity that was not shipped.
Warehouses	Subscription	RMS	These are messages that are communicated by RMS that contain warehouses set up in the system (RMS). SIM only gets physical warehouse records.
Vendor	Subscription	RMS, external (financial)	These are messages communicated by RMS containing vendors set up in the system (RMS or external financial system).

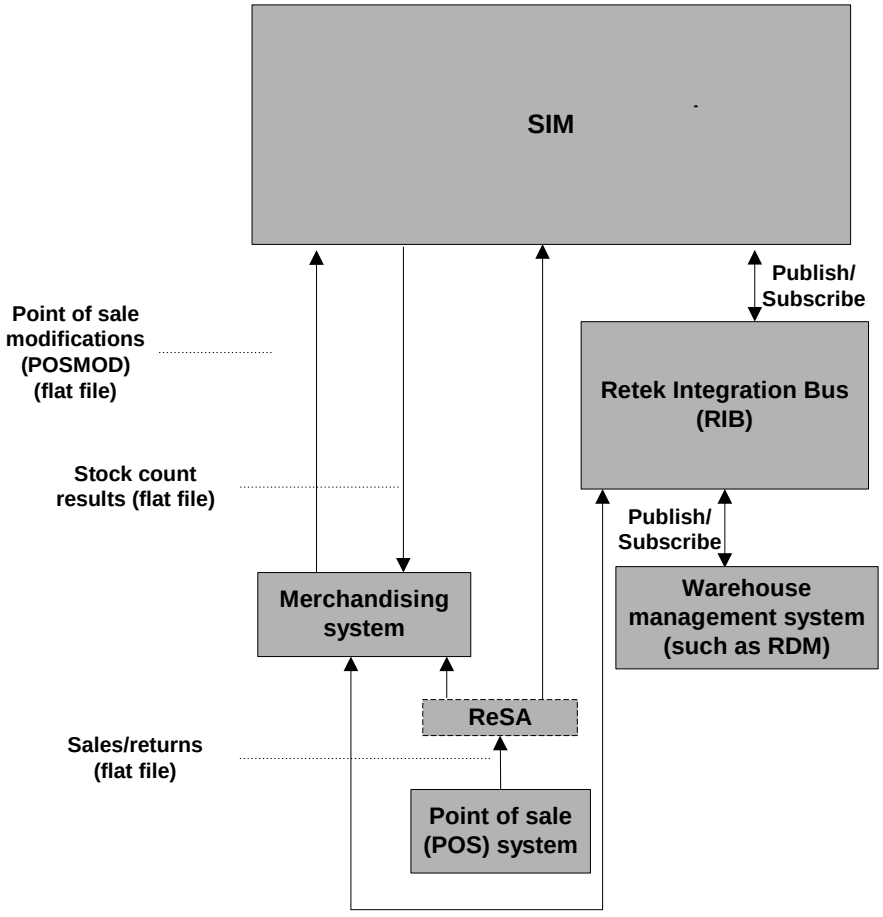
Chapter 5 – Integration interface dataflows

This chapter provides a functional overview of how SIM integrates with other systems (including other Retek systems).

Overview

The first section in this chapter provides you with a diagram illustrating the various Retek products and databases that SIM interfaces with as well as the overall dataflow among the products. The accompanying explanations are written from a system-to-system perspective, illustrating the movement of data.

System to system SIM dataflow



SIM-related dataflow across the enterprise

From SIM to the warehouse management system (WMS)

- Returns to warehouse
  Via the RIB, SIM sends outbound ASN data to facilitate the communication of store-to-warehouse shipment data to the WMS.

From the WMS to SIM

The following WMS data is published via the RIB for SIM subscription:

- Outbound advance shipping notice (ASN) data converted to inbound ASN data
  To facilitate warehouse-to-store shipments, the WMS provides SIM outbound ASN data. ASNs are associated with shipments and include information such as to and from locations, container quantities, and so on. Note that outbound ASN data is converted to inbound ASN data by the RIB for SIM's subscription purposes. The data is the same, but the format is slightly different. The conversion takes place so that ASN inbound data can be the same among applications.

From a point of sale (POS) system to SIM

The following data is sent from a point of sale (POS) system through ReSA (optional) to SIM via a flat file:

- Sales and returns data
  SIM uses the data to update the SOH for the store/items combinations in each file. In other words, SIM learns about inventory movement (what is sold and what is returned).

From the merchandising system to SIM

The following merchandising system data is published via the RIB for SIM subscription:

- PO data
  SIM allows the user to receive against direct store delivery (DSD)-related PO data. DSD occurs when the supplier drops off merchandise directly in the retailer's store.
- Item data (sellable and non-sellable items)
  SIM processes only transaction-level items (SKUs) and below (such as UPC), so there is no interface for parent (or style) level items. See the RMS documentation for more information about its three-level item structure. In addition to approved items records, the item data includes including header information, item/supplier, and item/supp/country details. Merchandise hierarchy data is an attribute of the item data to which SIM subscribes.
- Location data (updated store and warehouse location information)
- Item-location data
  SIM uses this data for ordering parameters (for example, allowing the user to determine whether an item is a store order type item).
- Diff data
- Supplier and supplier address data
- Transfer request data
  Corporate users can move inventory across stores via RMS transfer requests.

- Price change confirmation data
  In response to SIM's request, the merchandising system returns an approved or rejected status. If the request is approved within the merchandising system after it has performed its validation steps, the price change is sent to SIM through the POSMOD flat file.
- Return requests
  The merchandise system sends return requests from a store to a warehouse (RTW) and/or to a vendor (RTV). The store itself ships the goods.

From the merchandising system to SIM via the POSMOD module

Point-of-sale modifications (POSMOD) refers to a flat file which tracks any item changes made at the location level in the merchandising system. The POSMOD file is generated at the end of each day and is fed to SIM via a batch process. The following data is sent from the merchandising system to SIM via a flat file:

- Item-location association and price data
  SIM uses this information for reference/lookup purposes (for example, users can look up an item in the GUI and research its price at that location) and for all price changes.

From SIM to the merchandising system

The following SIM data is published via the RIB for the subscription of the merchandising system:

- Receipt data
  By sending the receipt data, SIM notifies the merchandising system of what SIM received. Types of receipt data are related to the following:
  - Transfers
  - Existing (merchandising system) POs associated with DSDs
  - New POs associated with DSDs
  - Merchandising system (such as RMS) purchase orders
- RTV and RTW data
  SIM notifies the merchandising system about returns to vendors and returns to warehouses.
- Return to warehouse data
  SIM uses ASN out data to notify the merchandising system about returns to warehouses.
- Item request data
  SIM sends this data to communicate a request for inventory of a particular item.
  - Store ordering data
    Store ordering is an extension of item requests using identical interfaces but different internal processes. The merchandising system uses this data to calculate a 'store order' replenishment type item's recommended order quantity (ROQ).
- Stock count schedule data
  The merchandising system uses this data to help maintain the synchronicity of the inventory levels in SIM and the merchandising system. Once the merchandising system has the stock count schedule data, SIM and the merchandising system perform a snapshot count at the same time. The store does a physical count and uploads the results, and the merchandising system compares the discrepancies.

- Price change request data
  A SIM user is able to request price changes, along with effective dates, from the merchandising system.

From SIM to the merchandising system via the stock upload module in the merchandising system

- Stock count results
  Once a stock count is authorized and completed, SIM creates a flat file and stages it to a directory. Using the flat file generated by SIM, the merchandising system's stock upload module retrieves and uploads the physical stock count data. The merchandising system uses this data to help maintain the synchronicity of the inventory levels in SIM and the merchandising system.

Chapter 6 – Functional design and overviews

This chapter provides information concerning the various aspects of SIM's functional areas. Topics include:

- Functional overviews
- A summary of the batch process(es) associated to the component
- The functional reasons for message subscriptions and publications

Note: For information about the functional reasons for foundation data-related RIB messages, see the section, 'Functional descriptions of messages' in '"Chapter 4 – SIM and the Retek Integration Bus (RIB)".

Inventory adjustments functional overview

To assist in maintaining perpetual inventory, SIM provides the ability to create inventory adjustments for all items within a store. SIM conveys changes to the merchandising system. Inventory adjustment functionality within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods.

Inventory adjustment processing within SIM includes the following features:

- Reason codes, which correspond to dispositions, can be assigned to inventory adjustments, thus moving stock to various inventory buckets. This code not only is used for reporting purposes, but also indicates to the system whether the amount is to be incremented or decremented and in which direction. Two examples follow:

  Example 1:
  A reason code of 'Removed for repair' would indicate to the system that the inventory for the selected item is to be decremented from the available SOH bucket and incremented in the unavailable SOH bucket.

  Example 2:
  A reason code of 'Return from repair' instructs the system to move the selected inventory by decrementing the unavailable SOH bucket and incrementing the available SOH bucket.

- Multiple inventory buckets are not only used to indicate to a store user what is available to sell, but this information also updates the central replenishment system ensuring proper ordering for all the stores.
- Manual adjustments can be made to the SOH inventory level for an item.
- Adjustments can be placed into pending status, which moves the stock to an unavailable bucket. Later, these adjustments can be transformed into regular inventory adjustments.
- The system notifies the merchandise system of all inventory adjustments.

A summary of reason codes and dispositions

The following table (shown for example purposes) provides a list of SIM's reason codes, their descriptions and the dispositions that are linked to the reason code. Note that retailers often configure the information on a database level to best fit their business needs. The abbreviations in the table correspond to the following names:

- A: Available
- O: Out
- U: Unavailable

For example, code 83 refers to a theft, which indicates that the stock is moved from available in the store to out (the stock is gone from the store).

Code	Reason code description	Disposition
1	Shrinkage	A -> O
81	Damage - Out	A -> O
82	Damage - Hold	A -> U
83	Theft	A -> O
84	Store Use	A -> O
85	Repair - Out	A -> U
3	Repair - In	U -> A
86	Charity	A -> O
87	Stock In	O -> A
88	Stock Out	A -> O
89	Dispose from on Hold	U -> O
90	Dispose from SOH	A -> O
91	Stock - Hold	A -> U
92	Admin	A -> O
93	Store Customer Return	O -> A
95	Consignment	A -> O
96	Ready to Sell	U -> A
97	Returns	U -> A
94	Product Transformation – In	O -> A
98	Product Transformation – Out	A - > O

Functional description of inventory adjustment-related RIB messages

The table below briefly describes the functional role that messages play with regard to inventory adjustment-related SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data's entry into the application's middle tier, see Chapter 4, "SIM and the Retek Integration Bus (RIB)". For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
Inventory adjustments	Publication	RMS	These messages are used by SIM to communicate inventory adjustments. RMS uses these messages to adjust inventory accordingly.

Item requests and store orders functional overview

Item requests

Item requests are used to send requests for goods to the merchandising system when the user notices there is a shortage or demand for particular items. The user creates the request and adds the items and the quantities to the item request. Note that item requests are restricted to any item that is *not* a store order replenishment item. Item request functionality within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods.

The item request is created in SIM. Item request data is published for the merchandising system's subscription. The merchandising system can attempt to generate either a purchase order or a warehouse delivery.

Store orders

Store ordering functionality is very similar to item request functionality. However, unlike item request functionality, which is only valid for items that are *not* associated to the store order replenishment process, store order functionality is valid *only* for items that are on the store order replenishment process.

Store orders can be created in two ways, scheduled through product group-related screens (available on the PC only) and through specific manual store order-creation screens (available on both the PC and the wireless device).

Store order creation through product group-related screens

Store orders (which must be divided by department) are created through the following high-level steps:

- Individual items and item hierarchies are associated into a single unit product group for the purpose of scheduling a store order.
- Product groups can be scheduled for a store order on a specified day or on scheduled intervals (for example, daily, weekly, monthly, or annually).
- A batch process generates the store order (with items and quantities) in a pending status. See the section, 'Store order generation batch process' later in this chapter.
- The user is able to enter the pending store order generated via the schedule and edit the items and quantities on the store order. Once the store order is requested, it is sent to the merchandising system where it will be fulfilled by either a PO or a warehouse-to-store transfer.

Manual store order creation

- The user is able to create store orders by adding items to the store order that are store order replenishment items for the same department.
- Once the store order is requested, it is sent to the merchandising system where it is fulfilled by either a PO or a warehouse-to-store transfer.

Store order generation batch process

The batch process looks for those product groups that are set up as 'store order type product group' that are scheduled for that day. The batch process generates the store order (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 PO(s) or warehouse to store transfer(s) as applicable.

For more information about batch processing within SIM, see Chapter 7, "Java batch processes".

Functional description of item request and store ordering-related RIB messages

The table below briefly describes the functional role that messages play with regard to item request and store ordering SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data's entry into the application's middle tier, see Chapter 4, "SIM and the Retek Integration Bus (RIB)". For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
Inventory request	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item. RMS uses this data to fulfill the requested inventory through either auto-replenishment or by creating a one-off purchase order/transfer.
Store ordering	Publication	RMS	These messages are used by SIM to communicate the request for inventory of a particular item. This functionality represents an extension of item request functionality and uses the same e*ways.

Price changes functional overview

Price changes are used by the retailer to change the price of a particular item at a location. Price changes are performed only on the PC. Within SIM, the following areas of functionality are related to price changes:

- SIM receives price changes, along with their effective dates, from the merchandising system through the POSMOD flat file. See “Appendix B – Batch file layout specifications” later in this document.
- A SIM user is able to request price changes, along with effective dates, from the merchandising system. SIM only allows items attached to a store level zone group to be added to a price change. In response to SIM’s request, the merchandising system returns an approved or rejected status on the price change request. If the request is approved within the merchandising system after it has performed its validation steps, the price change is sent to SIM through the POSMOD flat file.

Price changes from the merchandising system automatically become active in SIM when they reach the effective date.

Users can send price change items to the ticketing dialog to create item tickets or shelf edge labels. See the section, ‘Ticketing functional overview’, in this chapter.

A price change is included in the POSMOD flat file ‘x days’ prior to the price change’s effective date, where ‘x days’ represents a configuration held at the system level in the merchandising system.

Price changes-related batch processes

PosmodDnldFileParser

This batch process imports price data and item-location associations. Valid price change types are permanent, clearance, and promotion. A price change is included in the POSMOD flat file ‘x days’ prior to the price change’s effective date, where ‘x days’ represents a configuration held at the system level in the merchandising system. Once a new price change enters the SIM system, it is placed into ‘in progress’ status.

SIM’s price history table is affected. For example, a new record is entered into the table when SIM receives data that a price change has occurred. SIM’s item-location table is affected. For example, when a store is going to be selling an item for the first time, SIM enters a new record into its item-location table.

For more information about batch processing within SIM, see Chapter 7, “Java batch processes”.

ActivatePriceChanges

This batch process takes ‘pending’ or ‘ticket list’ price changes from the merchandising system (type = price change) that are effective for tomorrow and sets them to ‘active’ status.

If a new price change is going from pending to active, the batch determines whether the item has an existing price change as active. If so, the batch process changes that status to ‘completed’.

For more information about batch processing within SIM, see Chapter 7, “Java batch processes”.

Functional description of price changes-related RIB messages

The table below briefly describes the functional role that messages play with regard to price changes SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data’s entry into the application’s middle tier, see Chapter 4, “SIM and the Retek Integration Bus (RIB)”. For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
Price change confirm	Subscription	RMS	These messages contain a confirmation for a price change request. SIM uses this type of message to update the status of a price change created in SIM to reflect the price change’s status created by RMS.
Price change request	Publication	RMS	These messages contain a request for a price change. RMS will use this message to create a price change, then perform conflict checking against the price change, and return a confirmation to SIM including the status of the price change.

Receiving functional overview

Within SIM, the following areas of functionality are related to receiving:

- Transfers receiving
- Warehouse delivery
- Direct store delivery (DSD)

Transfers receiving

Because of transfers receiving-related processing within SIM, the retailer is able to receive against transfers on both the handheld device and the PC. The retailer can receive a store-to-store transfer by scanning an item on the transfer. By scanning and/or manually entering the data, the retailer adds the items to be received at the item or case level. Damaged items may be recorded during the receiving process. Once the transfer is received, it may be completed or saved to be completed at a later time. When the transfer receipt is completed, SIM decrements the inventory from in-transit status and increments the SOH inventory, as applicable.

Transfer receiving-related processing within SIM includes the following:

- Stock can be differentiated into different buckets depending on stock status (for example, unavailable stock, stock in transit, and so on).
- The system can automatically update stock inventory on the basis of the status of the transfer.
- The system can send emails automatically to the sending store in cases of damaged goods and/or discrepancies.
- The system can receive unexpected cases and items (configuration).
- Auto-receive functionality allows the retailer to set up stores in SIM that the retailer wishes to automatically receive transfers from. Auto-receive automatically adjusts the stock on hand of the receiving store when the listed store dispatches the transfer. Note that the receiving store is not able to receive exceptions, or record damages to the transfer.
- An inventory adjustment record is written for damaged units (with a reason code of ‘damaged’) to adjust the units out of SOH in the receiving store.
- Depending upon system configurations, users can re-open already received transfers and adjust received quantities (within an established number of days). Corrected data is then processed and resent to the merchandising system. This unit receiving adjustment functionality is only available on the PC.

Transfers that are under or over received

Transfers over received

When items are received for more than the dispatched quantity, the system adjusts the difference out of the sending store's SOH. No inventory adjustment record is sent to the merchandising system or displayed. An email notification of the adjustment is sent to the sending store. The email includes the transfer number, item number(s), and the quantities adjusted out.

Transfers under received (shortages)

When items are received for less than the dispatched quantity, the system adjusts the difference in the sending store's SOH. No inventory adjustment record is sent to the merchandising system or displayed. An email notification of the adjustment is sent to the sending store. The email includes the transfer number, item number(s), and the quantities adjusted in.

Warehouse delivery

Warehouse delivery functionality within SIM is utilized when goods are sent from a warehouse to a receiving store. Warehouse delivery within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods. SIM allows for receiving from any number of company-operated warehouses. The warehouses must be approved shipping locations for the receiving store, and the items shipped must be approved for delivery to the receiving store. Once items are received and the receipt is confirmed, SIM updates the status of the items from 'in-transit' from the distribution center to the store, to a status of 'on-hand'. The SOH is thus increased by the amount received.

Because of receiving-related functionality within SIM, the retailer can perform the following:

- Receive at the shipment, container, and case level.
- Ensure that containers not received are automatically placed into missing status.
- Perform a container lookup within the lookup dialogue to find out details about the warehouse delivery.
- Receive unexpected cases and items (configuration).

Additional functionality includes the following:

- An inventory adjustment record is written for damaged units (with a reason code of 'damaged') to adjust the units out of SOH in the receiving store.
- Depending upon system configurations, users can re-open warehouse deliveries and adjust received quantities (within an established number of days). Corrected data is then processed and resent to the merchandising system. This unit receiving adjustment functionality is only available on the PC.

Direct store delivery (DSD)

DSD occurs when the supplier drops off merchandise directly in the retailer's store. This process is common in convenience and grocery stores, where suppliers routinely come to restock merchandise. In these cases, the invoice may or may not be given to the store (as opposed to sent to corporate), and the invoice may or may not be paid for out of the register. The SIM system allows for the retailer to create new delivery records. This process allows the retailer to enter/scan a product bar code from any of the items in the delivery. Once the system verifies the bar code, the retailer can choose the supplier for the delivery. The system allows the retailer to either select an existing purchase order associated with that supplier to receive against, or to create a new purchase order. DSD delivery functionality within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods.

The retailer can enter invoice information and receive items by the case/pack or by the item. The retailer can also print a delivery receipt once all items have been received, and the delivery is finalized. Note that the system is also able to handle deliveries partially received, allowing for multiple receipts against a single PO.

Upon completing the delivery, the SOH for the store is updated with the received quantities. An inventory adjustment record is written for damaged units (with a reason code of ‘damaged’) to adjust the units out of SOH in the receiving store.

The receipt and purchase order information is published to RIB for the purposes of the merchandise system.

Depending upon system configurations, users can re-open direct deliveries and adjust received quantities (within an established number of days). Corrected data is then processed and resent to the merchandising system. This unit receiving adjustment functionality is only available on the PC.

Receiving against advanced shipment notices (ASN)

Because of receiving-related processing within SIM, the retailer is able to receive against advanced shipment notices (ASN) on both the handheld device and the PC. ASNs that originate at the vendor are published to the RIB, and SIM subscribes to the data.

When a direct delivery is received, SIM checks for a corresponding open ASN against the PO.

Retailers are prompted as to whether they would like to apply the ASN to the delivery. If the ASN is applied, the shipped quantities from the ASN are applied to the quantity received for the direct delivery. If new items are included in the ASN but do *not* reside on the original PO, the items are added to the PO. Once the ASN is applied, the retailer can modify any of the received quantities.

Direct EXchange (DEX) and Network Exchange (NEX) receiving

Direct EXchange (DEX) and Network Exchange (NEX) are uniform communications standards. DEX is the means through which a supplier, using a hand-held computer, can exchange electronic invoicing information with a store's direct store delivery (DSD) system. NEX differs in its delivery system, using the web as opposed to a hand-held cradle.

SIM is designed to support the integration of a supplier's DEX/NEX information into direct delivery-related screens, thereby simplifying the receiving process. Data is transferred to a store's DSD system using the Electronic Data Interchange (EDI) transaction set 894 (delivery/return base record). With the uploaded data, the store user can view, edit, and confirm the information contained in the file before receiving the direct delivery.

DEX/NEX-related batch process

This batch process imports the data within a direct delivery shipment (item, supplier, receiving location, and so on) from a file in the DEX/NEX directory (`dexnex`). The batch process opens the file to be read, parses the file, and inserts the PO, shipment, and receipt data into the database.

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.

For more information about batch processing within SIM, see Chapter 7, "Java batch processes".

Functional description of receiving-related RIB messages

The table below briefly describes the functional role that messages plays with regard to SIM's receiving-related functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data's entry into the application's middle tier, see Chapter 4, "SIM and the Retek Integration Bus (RIB)". For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
DSD receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of a supplier delivery for which no RMS purchase order had previously existed.
Purchase orders	Subscription	RMS	These messages contain approved, direct to store purchase orders. SIM uses these to receive direct deliveries against.

Functional area	Subscription/ publication	Integration to Products	Description
Receipts	Publication	RMS	These messages are used by SIM to communicate the receipt of an RMS purchase order, a transfer, or an allocation.
Receiver unit adjustments	Publication	RMS	These messages are used by SIM to communicate any adjustments to the receipts of purchase orders, transfers, and allocations.

Returns and return requests functional overview

Returns

SIM allows a store user to look up, create, edit, delete and complete returns from the store to a company owned warehouse and/or directly to the vendor. Returns functionality within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods.

Item quantities can be entered in eaches or cases. Once the applicable quantities are entered, the user is prompted to enter a reason code for the return. The reason codes are retailer defined. Available SOH is decremented for the return except when the item has unavailable inventory. In that case, the user is asked whether he or she would like to use that for the return. After the reason code is selected, the user may either complete the return or save it to be completed at a later date.

Once the return is completed, a return document can be printed to be used both as a report and/or a packing slip for the shipment.

Return requests

Return requests functionality return requests to be fulfilled from a store to a warehouse (RTW) and/or to a vendor (RTV) that were generated via the merchandising system.

A return request might be generated by a safety concern (for example, glass shards are discovered in a product). The functionality can be summed up as a return generated by the merchandising system that can be edited and approved in SIM. Return requests functionality within the SIM system is accomplished only on the PC-based deployment.

Once SIM receives the return request data, it ‘takes over’ the request and allows store users to add, edit, delete, save and dispatch the return request. Once the request is deleted or dispatched, a message is sent back to the merchandising system.

Functional description of return and return request-related RIB messages

The table below briefly describes the functional role that messages play with regard to return and return request-related SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data's entry into the application's middle tier, see Chapter 4, "SIM and the Retek Integration Bus (RIB)". For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
Return to vendor	Publication	RMS	These messages are used by SIM to communicate the shipment of a return to vendor from the store.
RTV request	Subscription	RMS	These are messages communicated by RMS that contain a request to return inventory to a vendor.
Stock order status	Publication	RMS	These messages are used by SIM to communicate the cancellation of any requested transfer quantities. For example, the merchandising system can create a transfer request for 90 units from a store. If the sending store only ships 75, a cancellation message is sent for the remaining 15 requested items.
Transfer request	Subscription	RMS	These messages are communicated by RMS and contain a request to transfer inventory out of a store. Upon shipment of the requested transfer, SIM uses the ASN out message to communicate what was actually shipped. In addition, SIM uses the stock order status message to cancel any requested quantity that was not shipped.

Sequencing functional overview

Sequencing functionality provides users the ability to know the relative location of an item in a store. Sequencing a store improves store processes and reduces the time that employees spend looking for items (during a stock count, for example). The retailer can sequence all items in the store and create unique locations to hold the items. The system can prompt users to a specific location to look for a specific item. Sequencing functionality within the SIM system can be accomplished on a PC based deployment, on a wireless handheld device, or on a combination of the two deployment methods.

Sequencing functionality includes three means by which items can be assigned to places within a store: Macro sequencing, fixture IDs, and micro sequencing. When ordering, the system follows this pattern.

Macro sequences represent the highest level of locations that are set up in the store. The user can create macro locations, assign items to macro locations, remove items from macro locations, move items within macro locations and resequence an entire macro location (wireless only).

A user can assign multiple fixture IDs to an item in a location. Fixture IDs allow users to attach any particular ID/label/letter to a sequenced item.

A micro sequence is the lowest (most granular) item location level. For example, a bin with a single fixture ID could include five micro-sequenced items.

The following table provides an example of sequencing:

Macro sequence	Fixture ID	Micro sequence
Produce	1	Oranges
	2	Apples
	3	Bananas
	4	Oranges
Frozen foods	A	TV dinners
	A	Burritos
	B	Burritos
Cereal	Shelf 1	Toasted oats
	Shelf 2	Toasted oats

Shelf replenishment functional overview

The replenishment process attempts to ensure that the shop floor inventory is set at a level best suited for customers.

Shelf replenishment functionality within SIM is related to the movement of goods from the back room to the shop floor. For example, when a user sees that a certain soda quantity is low, he or she can instigate a replenishment process so that more of the soda is moved from the back room.

Shelf replenishment-related processing within SIM includes the following features:

- The system calculates what should be held on the shop floor to ensure that customers' expectations of availability are maintained. Store employees are driven to replenish the most urgently needed items first.
- The system offers a display of the location from which stock can be picked and provides the container number to ease the search for stock when the pick lists are being filled.
- The system allows picking requests to be generated on demand.
- The system leads the user around the back room in micro sequence order, so that items can be picked in the most efficient matter.

Replenishment requires that the available SOH be divided into three buckets: shop floor, backroom, and delivery bay. Because of these buckets, shelf replenishment affects almost every area in the application. For example, inventory adjustments and transfers are affected because the system must take the inventory buckets into account when engaging in these areas of functional processing. The system's 'available' inventory is the sum of the three buckets.

When merchandise becomes available (enters the store through a transfer, a DSD, and so on), the merchandise is always placed in the backroom bucket.

When merchandise becomes unavailable (leaves the store through a transfer, becomes damaged, and so on), the system always processes the inventory adjustment in the following order:

1 Backroom

2 Shop floor

3 Delivery bay

Replenishment calculation summary

Once the calculation is started, the system engages in the following processing:

- The system gets all the items that are sequenced on the shop floor with capacity in the product group.
- If a previous pick list exists for the selected group and it is in progress, the system assumes that all of the items on the pick list will be completed. If a previous pick list exists and is *not* started, the system deletes the old pick list and creates a new one.
- The system checks and gets the configuration parameters established through the GUI (group unit of measure, fill percentage, and so on).
- The system gets the shelf quantities and available SOH for the items found.
- The system converts the quantities to the correct group default unit of measure.

- The system compares the shelf quantity to the summed shop floor capacity for every item to determine the percentage the item is out of stock.
- Once the 'out of stock' percentage is calculated for every item, the system orders the items from the highest out of stock percentage to the lowest. If any of the items have the same out of stock percentage, the system uses the item that has the least amount on the shelf. For example, if item A has 10 out of 100 on the shelf, and item B has 1 out of 10 on the shelf, they both have the same out of stock percentage. However, the system considers item B a higher priority because there is less of it on the shelf.
- For each item, the system calculates the pick amount that should be brought from the back room/delivery bay to the shop floor. Keeping the items in priority order, the system looks at the available SOH, the items in the back room/delivery bay, and to what percentage the shop floor needs to be filled. The system 'takes' the inventory from the back room first and then 'takes' the inventory from the delivery bay. The shop floor quantity can only be equal or less than the capacity.
- If the pick list type is within day, the system stops when the amount to pick is equal to the summed pick amount calculated by the system.
- If the pick list type is end of day, the system continues until all of the items are completed.
- If the system generated pick amount is a decimal, the system rounds down to the nearest whole number.
- The system generates and displays the list in sequence order to the user. If the items are not sequenced in the backroom, the system displays the items in item ID order.

Clean up pick list end of day batch

The end of day batch process runs at the end of each day to reset the delivery bay and close any open pending pick lists. The system takes the entire inventory from the delivery bay and moves it to the back room. Any pending or in progress pick lists are changed to a cancelled state. Users who are actioning a pick list are 'kicked out' of the system. That is, the system takes over their database lock, so they cannot make a save. After the batch process is run, all pick lists are either completed or cancelled, and the delivery bay has zero inventory.

For more information about batch processing within SIM, see Chapter 7, "Java batch processes".

Stock counts functional overview

SIM provides the ability to schedule, perform, and authorize stock counts. SIM includes the following four types of stock counts, each of which is described in this section:

- Ad hoc (wireless only)
- Unit only
- Unit and amount
- Problem line

Note that the counting processing is identical among the unit only, unit and amount, and problem line stock count types. What differs among these three stock count types is either the setup or the items being counted.

Portions of the stock count functionality, such as the setup of product groups, schedules, and authorizations, are performed on the PC only. The actual counting of inventory can be performed on both the PC and the wireless device. Wireless stock counts are divided so that each user of a handheld device performs a stock count according to a macro location (for example, vegetables, cereals, and so on).

Ad hoc stock counts

An ad hoc stock count is performed by a user walking through the store scanning any items that need to be counted. Ad hoc stock counts are only performed on the handheld side of the system. They have no group or schedule functionality associated with them.

Ad hoc stock count processing within SIM includes the following features:

- The system creates an inventory snapshot as each item is identified and added to the stock count.
- An ad hoc stock count can be saved and completed later; existing ad hoc stock count can be retrieved and resumed.
- The system utilizes discrepancy thresholds (established in administration setup by the retailer) based on a percentage or standard unit of measure by the item’s class in the merchandise hierarchy.
- On the PC, the system allows for the authorization of items that are discrepant (based on the percentage or standard unit of measure thresholds).
- Where the authorized item quantities entered by the user differ from the SOH, the system creates inventory adjustments that are sent to the merchandising system. See the section, ‘Inventory adjustments’, in this chapter.

Unit only stock counts

Unit only stock count processing within SIM includes the following features:

- Individual items and item hierarchies are associated into a single unit product group for the purpose of scheduling a stock count.
- Unit product groups can be scheduled for a stock count on a specified day or on scheduled intervals (for example, daily, weekly, monthly, or annually).
- One or more stores can be assigned to the scheduled stock count.
- A stock count item list is generated at the store level via a batch process (described below) that runs daily.
- Users with proper security are prompted of any discrepancies outside of set tolerances, and the system can automatically force a recount if the discrepancies are too high. The system utilizes discrepancy thresholds (established in administration setup by the retailer) based on a percentage or standard unit of measure by the item's class in the merchandise hierarchy.
- On the PC, the system allows for the authorization of items that are discrepant (based on the percentage or standard unit of measure thresholds).
- Where the authorized item quantities entered by the user differ from the SOH, the system creates inventory adjustments that are sent to the merchandising system. See the section, 'Inventory adjustments', in this chapter.

Unit and amount stock counts

Unit and amount stock count processing within SIM includes the following features:

- Item hierarchies are associated into a single unit and amount product group for the purpose of scheduling a stock count.
- Unit and amount product groups can be scheduled for a stock count on a specified day.
- The stock count schedule for unit and amount stock counts is sent to the merchandising system anytime it is created/updated/deleted.
- One or more stores can be assigned to the scheduled stock count.
- A stock count item list is generated at the store level via a batch process (described below) that runs daily.
- Users with proper security are prompted of any discrepancies outside of set tolerances, and the system can automatically force a recount if the discrepancies are too high. The system utilizes discrepancy thresholds (established in administration setup by the retailer) based on a percentage or standard unit of measure by the item's class in the merchandise hierarchy.
- On the PC, the system allows for the authorization of items that are discrepant (based on the percentage or standard unit of measure thresholds).
- Where the authorized item quantities entered by the user differ from the SOH, a flat file is sent to the merchandising system with a header that contains the stock count ID, stock count date, and the store number that executed the count. The file contains details for each item and the quantity counted. This information is then staged in the merchandising system and can be used for reporting purposes.

- The actual counted values for merchandise and the booking numbers can be consolidated at year end.

Problem line stock counts

This functionality gives stores the ability to create automated stock counts according to predefined criteria (for example, the retailer could decide to count all of the items that have negative SOH values).

Once stores have established the criteria (based upon problematic areas), a batch process runs to find any items that meet the criteria. The 'found' items are added to a system-generated stock count. They are counted in the same way as in a scheduled stock count. Note that problem line stock counts have no schedule functionality associated with them.

Problem line stock count processing within SIM includes the following features:

- Individual items and item hierarchies are associated into a single problem line product group.
- A problem line stock count item list is generated at the store level via a problem line batch process that is based on problem line parameters. The batch process is described below.
- Users with proper security are prompted of any discrepancies outside of set tolerances, and the system can automatically force a recount if the discrepancies are too high. The system utilizes discrepancy thresholds (established in administration setup by the retailer) based on a percentage or standard unit of measure by the item's class in the merchandise hierarchy.
- On the PC, the system allows for the authorization of items that are discrepant (based on the percentage or standard unit of measure thresholds).
- Where the authorized item quantities entered by the user differ from the SOH, the system creates inventory adjustments that are sent to the merchandising system. See the section, 'Inventory adjustments', in this chapter.

Stock count generation batch processes

Unit and unit and amount

On a daily basis, a batch process creates the stock counts that are scheduled for the current day. The system looks at all the scheduled stock count records and determines whether any are scheduled for today. The process creates the stock counts for each individual store. If a scheduled count includes a list of 5 stores, then 5 separate stock count records are created.

For more information about batch processing within SIM, see Chapter 7, "Java batch processes".

Problem line

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.

In the chance that 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 might 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.

For more information about batch processing within SIM, see Chapter 7, “Java batch processes”.

Functional description of stock count-related RIB messages

The table below briefly describes the functional role that messages play with regard to stock count-related SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data’s entry into the application’s middle tier, see Chapter 4, “SIM and the Retek Integration Bus (RIB)”. For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
Stock count schedules	Publication	RMS	These messages are used by SIM to communicate unit and value stock count schedules to RMS. RMS uses this schedule to take an inventory snapshot of the date of a scheduled count.

Ticketing functional overview

This version of SIM provides a single, generic report for shelf labels and item tickets that is intended for printing demonstration only. The report contains information such as item, item description, price, and so on. A future release of SIM will include an integrated reporting tool to allow users to select different printing formats for actual ticket and label printing.

Ticketing is internal SIM functionality that will allow the user to print shelf edge labels and item tickets for stock.

Label formats and label quantities are held for items set up in sequencing. Entire macro locations (for example, produce) can be printed from within sequencing screens. For more information about sequencing, see the section, ‘Sequencing functional overview’ in this chapter.

An item tickets dialog allows users to generate tickets for a specific item on the PC and the wireless device.

Tickets can be generated from price changes and from purchase orders (PO) that have been received.

Transfer out functional overview

Within SIM, the following areas of functionality are related to transfers and are discussed in this section:

- The creation of store to store transfers
- The creation of store to store transfer requests
- Email alerts

The creation of store-to-store transfers

SIM allows for the lookup, creation, editing, and deletion of store-to-store transfers. A store-to-store transfer is the movement of stock from one store to another, within a given company.

This functionality can be performed on the PC deployment, on a wireless device, or a combination of both. Users can create a transfer by selecting the receiving store and adding items by scanning and/or engaging in manual entry. The system verifies the receiving store is approved to receive the selected items and that the sending store has the available stock on hand (SOH) inventory. A transfer can be immediately sent or saved to be dispatched at a later time. At the point the transfer is dispatched, SIM decrements the onhand inventory from the sending store and increments the in-transit inventory for the receiving store.

Note: For information about transfer receiving, see the section ‘Receiving functional overview’ in this chapter.

Note the following features of transfer-related functionality within SIM:

- Stock is differentiated by different buckets depending on stock status (for example, in-transit stock, reserved for transfer stock, and so on).
- The system automatically updates stock inventory on the basis of the status of the transfer.
- Buddy store functionality allows for the setup of a group of stores within a transfer zone in SIM to which the retailer often transfers items. This shortens the list of values that users select from when they create a transfer. Note that the retailer continues to have the option of creating a transfer to any store outside of the buddy store group, as long as it resides within the transfer zone.

An overview of stock movement after a successful dispatch

1 The stock moves from the transfer reserved and transfer expected buckets.

2 The transfer reserved quantity for the outbound location decreases as does the SOH for the outbound location.

3 The transfer expected for the receiving store results in a stock movement to the in transit bucket. The transfer quantity is removed from the in transit bucket to the SOH bucket when the receiving store receives the transfer.

Transfer requests

Transfer requests provide stores the ability to ask for products from other stores or allow corporate users to move inventory across stores via merchandising system transfer requests. The following processing occurs:

1 The transfer request is initiated from or on the behalf of a requesting store.

2 The store receiving the request either accepts or rejects the transfer request.

3 If accepted, the transfer is created, and the transfer of goods takes place from the store that received the transfer request to the requesting store.

Retailers are only allowed to accept or reject a transfer request awaiting response on the PC. The actual transfer request can be created on either the PC or the wireless device.

Transfer-related email alerts batch process

The email alerts batch process determines how long transfers have been in a pending status and then sends an email alert depending upon date parameters established by the retailer. The batch process first checks all the dispatched transfers and the configurable number of days. If the program finds a dispatched transfer that has a dispatch date equal to or older than the number of days established, an email alert is sent to both the sending store and the receiving store.

For more information about batch processing within SIM, see Chapter 7, “Java batch processes”.

Functional scenarios of transfer-related RIB processing

Store-to-store transfer initiated in SIM by the sending store

The numbered explanation below pertains to a store-to-store transfer that is initiated in SIM by the sending store, where both stores are in SIM.

1 Internal SIM processing occurs.

2 SIM publishes ASNOUT message(s) upon the shipment of transfer (dispatch) to notify the merchandising system of the shipment.

3 SIM publishes the RECEIVING message(s) upon receipt of the transfer to notify the merchandising system of receipt.

Store-to-store transfer initiated in SIM by the receiving store

The numbered explanation below pertains to a store-to-store transfer that is initiated in SIM by the receiving store, where both stores are in SIM.

1 Internal SIM processing occurs.

2 SIM publishes the ASNOUT message(s) upon the shipment of transfer (dispatch) to notify the merchandising system of the shipment.

3 SIM publishes the RECEIVING message(s) upon receipt of the transfer to notify the merchandising system of receipt.

Store-to-store transfer initiated on a corporate level in the merchandising system

The numbered explanation below pertains to a store-to-store transfer that is initiated on a corporate level in the merchandising system.

1 Upon the creation of the transfer, the merchandising system publishes a STOCKORDER message(s).

2 Internal SIM processing occurs. Because SIM subscribes to the transfer request originally published from the merchandising system, SIM publishes a response to the merchandising system via a SOSTATUS (stock order status) message, regardless of whether the user accepts or rejects the transfer request.

3 The following messages are associated with the shipment of the transfer:

   a SIM publishes the ASNOUT message(s) upon the shipment of transfer (dispatch) to notify the merchandising system of the shipment.

   b SIM subscribes to ASNIN message(s) for shipment notification data. To limit the amount of code that needs to be maintained within SIM, the RIB converts ASNOOUT message(s) to the ASNIN message format to allow SIM to maintain a single subscriber regardless of the source of the ASN. In other words, warehouse deliveries and direct deliveries are communicated using the same message format (ASNIN).

   c The RECEIVING message(s) is sent upon receipt of the transfer to notify the merchandising system of the receipt.

Functional description of transfer-related RIB messages

The table below briefly describes the functional role that messages play with regard to transfer-related SIM functionality. The table also illustrates whether SIM is publishing the message to the RIB or subscribing to the message from the RIB. For the document type definition (DTD) that describes the XML message, and (in the case of subscriptions) the subscribing classes (injectors) that facilitate the data's entry into the application's middle tier, see Chapter 4, "SIM and the Retek Integration Bus (RIB)". For additional information, see the latest Retek Integration Guide and other RIB documentation.

Functional area	Subscription/ publication	Integration to Products	Description
ASN in	Subscription	RDM, Vendor (external)	These messages contain inbound shipment notifications from both vendors (PO shipments) and warehouses (transfer and allocation shipments).

Functional area	Subscription/ publication	Integration to Products	Description
ASN out	Publication	RMS, RDM	These messages are used by SIM to communicate store-to-warehouse transfers (returns to warehouse) to both RMS and RDM. These messages are also used to communicate store-to-store transfers to RMS.
Stock order status	Publication	RMS	These messages are used by SIM to communicate the cancellation of any requested transfer quantities. For example, the merchandising system can create a transfer request for 90 units from a store. If the sending store only ships 75, a cancellation message is sent for the remaining 15 requested items.
Transfer request	Subscription	RMS	These messages are communicated by RMS and contain a request to transfer inventory out of a store. Upon shipment of the requested transfer, SIM uses the ASN out message to communicate what was actually shipped. In addition, SIM uses the stock order status message to cancel any requested quantity that was not shipped.

Wastage functional overview

Wastage is the process through which inventory is lost over time (bananas turning black, for example).

In order to maintain more accurate inventory values, SIM's wastage functionality provides users in stores the ability to create wastage product groups. Variance percentage or standard UOM amounts can be set up on the wastage product group. Individual items and item hierarchies can be associated into a product group.

A user can schedule the date when a wastage product group batch process is run, and inventory adjustments are automatically made based upon the variances setup on the product group. Inventory adjustments are sent over the RIB to the merchandising system.

Wastage batch process

The batch process looks for those wastage product groups that are scheduled for today and creates an inventory adjustment (decrement) 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 passed to the merchandising system.

For more information about batch processing within SIM, see Chapter 7, "Java batch processes".

Chapter 7 – Java batch processes

This chapter provides the following:

- An overview of SIM's batch processing
- A description of how to run batch processes, along with key parameters
- A functional summary of each batch process, along with its dependencies
- A description of some of the features of the batch processes (batch return values, restart and recovery, and so on)

Batch processing overview

SIM's batch processes are run in Java. For the most part, batch processes engage in their own primary processing. However, there are some calls from the batch processes which utilize code from the normal services that are running in the server. Usually, this processing occurs when the batch processes engage in actions outside their primary processing (for example, when they utilize a helper method, touch the database, and so on).

Note the following characteristics of the SIM's 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 Java-based batch process

For Unix systems, Java processes are scheduled through executable shell scripts (.sh files). For Windows systems, Java processes are scheduled through executable batch files (.bat files).

Retek provides the shell scripts (.sh files) and batch files (.bat files). They perform the following internally:

- Set up the Java runtime environment before the Java process is run.
- Trigger the Java batch process.

For the DexnexFileParser, PosmodDnldFileParser, and the ResaFileParser batch processes, a parameter entered in the command line following the batch shell script/batch file name provides the batch process with the name and location of the file to process.

Summary of executable files associated to Java packages and classes

The following table describes the executable shell scripts, batch files, and Java packages along with the main class within them that defines the (batch) Java class that runs.

Executable shell script	Executable batch file for windows	Java package	Class
ActivatePriceChanges.sh	ActivatePriceChanges.bat	com.chelseasystems.cs.batch	ActivatePriceChanges
CleanupPickList.sh	CleanupPickList.bat	com.chelseasystems.cs.batch	CleanupPickList
DexnexParser.sh	DexnexParser.bat	com.chelseasystems.cs.dataaccess.ms.rms	DexnexFileParser
FrontEnd.sh	FrontEnd.bat	com.chelseasystems.cs.batch.frontend	FrontEnd
PosmodDnldFileParser.sh	PosmodDnldFileParser.bat	com.chelseasystems.cs.dataaccess.ms.rms	PosmodDnldFileParser
PurgeData.sh	PurgeData.bat	com.chelseasystems.cs.batch	PurgeData
ResaFileParser.sh	ResaFileParser.bat	com.chelseasystems.cs.dataaccess.resa	ResaFileParser
rssAlertScan.sh	rssAlertScan.bat	com.chelseasystems.cs.util	AlertDispatcher

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.

Functional descriptions and dependencies

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

Batch process	Details	Batch dependencies
ActivatePriceChanges	This batch process takes 'pending' or 'ticket list' price changes from the merchandising system (type = price change) that are effective for tomorrow and sets them to 'active' status. If a new price change is going from pending to active, the batch determines whether the item has an existing price change as active. If so, the batch process changes that status to 'completed'.	No dependencies
CleanupPickList	The end of day batch process runs at the end of each day to reset the delivery bay and close any open pending pick lists. The system takes the entire inventory from the delivery bay and moves it to the back room. Any pending or in progress pick lists are changed to a cancelled state. Users who are actioning a pick list are 'kicked out' of the system. That is, the system takes over their database lock, so they cannot make a save. After the batch process is run, all pick lists are either completed or cancelled, and the delivery bay has zero inventory.	No dependencies

Batch process	Details	Batch dependencies
DexnexFileParser	This batch process imports the data within a direct delivery shipment (item, supplier, receiving location, and so on) from a file in the DEX/NEX directory (`dexnex`). The batch process opens the file to be read, parses the file, and inserts the PO, shipment, and receipt data into the database. 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. See "Appendix B" later in this document for a file layout specification.	No dependencies
Frontend–'O' Note: The command line parameter following the batch process name determines what processing the batch process engages in. O = Store orders	Store orders The batch process looks for those product groups that are set up as 'store order type product group' that are scheduled for that day. The batch process generates the store order (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 PO(s) or warehouse to store transfer(s) as applicable.	No dependencies

Batch process	Details	Batch dependencies
Frontend–'P'  Note: The command line parameter following the batch process name determines what processing the batch process engages in. P = Problem line	Problem line 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. In the chance that 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 might 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.	No dependencies
Frontend–'S'  Note: The command line parameter following the batch process name determines what processing the batch process engages in. S = Unit or unit and amount	Unit or unit and amount On a daily basis, a batch process creates the stock counts that are scheduled for the current day. The system looks at all the scheduled stock count records and determines whether any are scheduled for today. The process creates the stock counts for each individual store. If a scheduled count includes a list of 5 stores, then 5 separate stock count records are created.	No dependencies

Batch process	Details	Batch dependencies
Frontend–'W' Note: The command line parameter following the batch process name determines what processing the batch process engages in. W = Wastage	Wastage The batch process looks for those wastage product groups that are scheduled for today and creates an inventory adjustment (decrement) 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 passed to the merchandising system.	No dependencies
PosmodDnldFileParser	This batch process imports price data and item-location associations. Valid price change types are permanent, clearance, and promotion. A price change is included in the POSMOD flat file 'x days' prior to the price change's effective date, where 'x days' represents a configuration held at the system level in the merchandising system. Once a new price change enters the SIM system, it is placed into 'in progress' status. SIM's price history table is affected. For example, a new record is entered into the table when SIM receives data that a price change has occurred. SIM's item-location table is affected. For example, when a store is going to be selling an item for the first time, SIM enters a new record into its item-location table. See "Appendix B" later in this document for a file layout specification.	No dependencies

Batch process	Details	Batch dependencies
PurgeData	The batch purging process deletes data from database tables while maintaining database integrity. 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).	No dependencies
ResaFileParser	This batch process imports sales and returns data that originates in a point of sale (POS) system. SIM uses the data to update the SOH for the store/items combinations in each file. In other words, from the batch process, SIM learns about inventory movement (what is sold and what is returned). Note that once SIM attains the data, it assumes that sales should be taken from the store shelf-related inventory bucket. This assumption is important to SIM's shelf replenishment processing. Similarly, SIM assumes that returns should go first to the backroom bucket, the system's logic is that returns must be inspected. See "Appendix B" later in this document for a file layout specification.	No dependencies
rssAlertScan	The email alerts batch process determines how long transfers have been in a pending status and then sends an email alert depending upon date parameters established by the retailer. The batch process first checks all the dispatched transfers and the configurable number of days. If the program finds a dispatched transfer that has a dispatch date equal to or older than the number of days established, an email alert is sent to both the sending store and the receiving store.	No dependencies

A note about multi-threading and multiple processes

SIM's batch processes are currently *not* set up to be multi-threaded or to undergo multi-processing.

A note about restart and recovery

Most SIM Java-based batch processes do not utilize any type of restart and recovery. Rather, if a restart is required, a batch process can simply be restarted. See the section 'Special processing for PosmodDnldFileParser and DexnexFileParser' later in this chapter for exceptions.

Special processing for PosmodDnldFileParser and DexnexFileParser

For information about logging in SIM for batch processes other than for the parsing engaged by PosmodDnldFileParser and DexnexFileParser, see "Chapter 2 – Backend system configuration".

Rerun file for PosmodDnldFileParser

If a failure occurs during the parsing processing of PosmodDnldFileParser, the batch process creates a rerun file in the same directory as the file being processed (`posmod`). The system places the line(s) from the original file associated with the error(s) into this rerun file. The rerun file name is based upon the original file's name.

The original file is automatically moved to a POSMOD originals directory (`posmodOriginals`) because the original file does *not* have to be processed again. The retailer can compare the rerun file to the original to determine the source of the issue. To run the rerun file, the operator should use the same command line that was used to run the original batch process but with the rerun file name specified in the command line.

Log file for PosmodDnldFileParser

A log file for the parsing portion of PosmodDnldFileParser is located in the same directory as the file being processed (`posmod`). The system logs the following two types of messages related to PosmodDnldFileParser batch processing:

- Warnings
  Warnings are *not* as serious as errors. They are logged for informational purposes, and they do not affect the running of the batch process.
- Errors
  Errors are the more serious of the two types of error messages. If these occur, they *do* affect the processing of the file. The line associated with the error is *not* processed but is instead placed into the rerun file.

Log file for DexnexFileParser

A log file for the parsing portion of DexnexFileParser is located in the `dexnexErrors` directory.

Appendix A – Stock count file layout specification

Stock count results flat file specification

Once a stock count is authorized and completed, the SIM server creates a flat file during runtime and stages it to a directory, which is configured during installation. Using the flat file generated by SIM, the merchandising system's stock upload module retrieves and uploads the physical stock count data. The file is formatted as follows:

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) YYYYMM DDHHMISS	date written by convert program
	stocktake_date	Date(14) YYYYMM DDHHMISS	stake_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. Ex: Back Stockroom or Front Window Display

Record name	Field name	Field type	Description
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.

Appendix B – Batch file layout specifications

Flat file used in the ResaFileParser batch process

The file below contains the sales and returns data that originates in a point of sale (POS) system and is sent through ReSA to SIM via a flat file. The file is formatted as follows:

Record Name	Field Name	Field Type	Default Value	Description	Required
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type	
	File Line Identifier	Char(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Type Definition	Char(4)	POSU	Identifies file as 'POS Upload'	Yes
	File Create Date	Char(14)	create date	date file was written by external system	Yes
	Location Number	Number(4)	specified by external system	Store or warehouse identifier	Yes
	Vat include indicator	Char(1)		Determines whether or not the store stores values including vat. Not required but populated by Retek sales audit	Yes
	Vat region	Number(4)		Vat region the given location is in. Not required but populated by Retek sales audit	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Currency code	Char(3)		Currency of the given location. Not required but populated by Retek sales audit	Yes
	Currency retail decimals	Number(1)		Number of decimals supported by given currency for retails. Not required but populated by Retek sales audit	Yes
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies transaction record type	
	File Line Identifier	Char(10)	specified by external system	ID of current line being processed by input file.	Yes
	Transaction Date	Char(14)	transaction date	date sale/return transaction was processed at the POS	Yes
	Item Type	Char(6)	'UPC' or 'SKU'	item type will be represented as a UPC, a SKU	Yes
	Item Value	Char(13)	item identifier	the id number of a SKU or UPC	Yes
	Supplement	Char(5)	supplemental identifier	used to further specify the id of a UPC item, or the pre-pack id reference	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	System_ind	Char(1)	'S' – staple sku 'f' – fashion sku 'P' – pack item	The type of item sold or returned. Not required but populated by Retek sales audit	Yes
	Dept	Number(4)	Item's dept	Dept of item sold or returned. Not required but populated by Retek sales audit	Yes
	Class	Number(4)	Item's class	Class of item sold or returned. Not required but populated by Retek sales audit	Yes
	Subclass	Number(4)	Item's subclass	Subclass of item sold or returned. Not required but populated by Retek sales audit	Yes
	Wastage Type	Char(6)	Item's wastage type	Wastage type of item sold or returned. Not required but populated by Retek sales audit	Yes
	Wastage Percent	Number(12)	Item's wastage percent	Wastage percent of item sold or returned with 4 implied decimal places. Not required but populated by Retek sales audit	Yes

Record Name	Field Name	Field Type	Default Value	Description	Required
	Transaction Type	Char(1)	'S' – sales 'R' - return	Transaction type code to specify whether transaction is a sale or a return	Yes
	Total Sales Quantity	Number(12)		Number of units sold at a particular location with 4 implied decimal places.	Yes
	Sales Sign	Char(1)	'P' - positive 'N' - negative	Determines if the Total Sales Quantity and Total Sales Value are positive or negative.	Yes
	Total Sales Value	Number(20)		Sales value, net sales value of goods sold/returned with 4 implied decimal places.	Yes
	Last Modified Date	Char(14)		For VBO future use	
Transaction Detail	File Type Record Descriptor	Char(5)	TDETL	Identifies transaction record type	
	File Line Identifier	Char(10)	specified by external system	ID of current line being processed by input file.	Yes
	Promotional Tran Type	Char(6)	promotion type – valid values see code_detail table.	code for promotional type from code_detail, code_type = 'PRMT'	Yes
	Promotion Number	Number(4)	promotion number	promotion number from the RMS	No

Record Name	Field Name	Field Type	Default Value	Description	Required
	Sales Quantity	Number(12)		number of units sold in this prom type with 4 implied decimal places.	Yes
	Sales Value	Number(20)		value of units sold in this prom type with 4 implied decimal places.	Yes
	Discount Value	Number(20)		Value of discount given in this prom type with 4 implied decimal places.	Yes
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type	
	File Line Identifier	Char(10)	specified by external system	ID of current line being processed by input file.	Yes
	Transactio n Count	Number(6)	specified by external system	Number of TDETL records in this transaction set	Yes
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type	
	File Line Identifier	Number(10)	specified by external system	ID of current line being processed by input file.	Yes
	File Record Counter	Number(10)		Number of records/transact ions processed in current file (only records between head & tail)	Yes

Flat file used in the PosmodDnldFileParser batch process

The single output file contains all records for all stores in a given run. The Retek standard file format is used, FHEAD, FDETL, FTAIL.

All input comes from the POS_MODS table in RMS.

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Identifier	Number ID(10)	Sequential number Created by program.	ID of current line being created for output file.
	File Type Definition	Char(4)	POSD	Identifies file as 'POS Download'
	File Create Date	Char(8)	Create date (vdate).	Current date, formatted to 'YYYYMMDD'.
File Detail	File Type Record Descriptor	Char(5)	FDETL	Identifies file record type
	File Line Identifier	Number ID(10)	Sequential number. Created by program.	ID of current line being created for output file.
	Location Number	Number(10)	Store	Contains the store location that has been affected by the transaction
	Update Type	Char(1)	Update type. Created by program.	Code used for client specific POS system. 1 - Transaction Types 1 & 2. 2 - Transaction Types 10 thru 18, 31 & 32, 50 thru 57, 59 thru 64. 3 - Transaction Types 21 & 22 4 - Transaction Types 25 & 26 0 - All other Transaction Types. These should never exist.

Record Name	Field Name	Field Type	Default Value	Description
	Start Date	Char(8)	Start_date or vdate + 1 if NULL.	The effective date for the action determined by the transaction type of the record. Formatted to 'YYYYMMDD'.
	Time	Char(6)	Start_time, End_time or start_date.	This field will be used in conjunction with starting a promotion (Transaction Type = 31). Start time will indicate the time of day that the promotion is scheduled to start. This field will also be used in conjunction with ending a promotion (Transaction Type = 32). Any other Transaction Type will use the time from the start_date column. Formatted to 'HH24MISS'.
	Transaction Type	Number(2)	Tran_type	Indicates the type of transaction to determine what Retek action is being sent down to the stores from the Retek pos_mods table. Valid values include: 01 - Add new transaction level item 02 - Add new lower than transaction level item 10 - Change Short Description of existing item 11 - Change Price of an existing item 12 - Change Description of an existing item 13 - Change Department/Class/Subclass of an existing item 16 - Put Item on Clearance 17 - Change existing item's Clearance Price 18 - Remove Item from Clearance and Reset 20 - Change in VAT rate

Record Name	Field Name	Field Type	Default Value	Description
				21 - Delete existing transaction level item 22 - Delete existing lower than transaction level item 25 - Change item's status 26 - Change item's taxable indicator 31 - Promotional item - Start maintenance 32 - Promotional item - End maintenance 50 - Change item's launch date 51 - Change item's quantity key options 52 - Change item's manual price entry options 53 - Change item's deposit code 54 - Change item's food stamp indicator 55 - Change item's WIC indicator 56 - Change item's proportional tare percent 57 - Change item's fixed tare value 58 - Change item's rewards eligible indicator 59- Change item's electronic marketing clubs 60 - Change item's return policy 61 - Change item's stop sale indicator 62 – Change item’s returnable indicator 63 – Change item’s refundable indicator 64 – Change item’s back order indicator

Record Name	Field Name	Field Type	Default Value	Description
	Item Number ID	Char(25)	Item	This field identifies the unique alphanumeric value for the transaction level item. The ID number of a item from the Retek item_master table.
	Item Number Type	Char(6)	Item_number_type	This field identifies the type of the item number ID.
	Format ID	Char(1)	Format_id	This field identifies the type of format used if the item_number_type is 'VPLU'.
	Prefix	Number(2)	Prefix	This field identifies the prefix used if the item_number_type is 'VPLU'. In case of single digit prefix, the field will be right-justified with blank padding.
	Reference Item	Char(25)	Ref_item	This field identifies the unique alphanumeric value for an item one level below the transaction level item.
	Reference Item Number Type	Char(6)	Ref_Item_number_type	This field identifies the type of the ref item number ID.
	Reference Item Format ID	Char(1)	Ref_Format_id	This field identifies the type of format used if the ref item_number_type is 'VPLU'.
	Reference Item Prefix	Number(2)	Ref_Prefix	This field identifies the prefix used if the ref item_number_type is 'VPLU'. In case of single digit prefix, the field will be right-justified with blank padding.
	Item Short Description	Char(20)	Item_short_desc	Contains the short description associated with the item.
	Item Long Description	Char(100)	Item_long_desc	Contains the long description associated with the item.
	Department ID	Number(4)	Dept	Contains the item's associated department.
	Class ID	Number(4)	Class	Contains the item's associated class.
	Subclass ID	Number(4)	Subclass	Contains the item's associated subclass.

Record Name	Field Name	Field Type	Default Value	Description
	New Price	Number(20)	New_price	Contains the new effective price in the selling unit of measure for an item when the transaction type identifies a change in price. Otherwise, the current retail price is used to populate this field. This field is stored in the local currency.
	New Selling UOM	Char(4)	New_selling_UOM	Contains the new selling unit of measure for an item's single-unit retail.
	New Multi Units	Number(12)	New_multi_units	Contains the new number of units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made.
	New Multi Units Retail	Number(20)	New_multi_units_retail	Contains the new price in the selling unit of measure for units sold together for multi-unit pricing. This field is only filled when a multi-unit price change is being made. This field is stored in the local currency.
	New Multi Selling UOM	Char(4)	New_multi_selling_UOM	Contains the new selling unit of measure for an item's multi-unit retail.
	Status	Char(1)	Status	Populates if tran_type for the item is 1(new item added) or 25 (change item status) or 26 (change taxable indicator). Contains the current status of the item at the store. Valid values are: A = Active I = Inactive D = Delete C = Discontinued

Record Name	Field Name	Field Type	Default Value	Description
	Taxable Indicator	Char(1)	Taxable_ind	Populates if tran_type for the item is 1 (new item added) or 25 (change item status) or 26 (change taxable indicator). Indicates whether the item is taxable at the store. Valid values are 'Y' or 'N'.
	Promotion Number	Number(10)	Promotion	This field contains the number of the promotion for which the discount originated. This field, along with the Mix Match Number or Threshold Number is used to isolate a list of items that tie together with discount information.
	Mix Match Number	Number(10)	Mix_match_no	This field contains the number of the mix and match in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items which tie together with the mix and match discount information.
	Mix Match Type	Char(1)	Mix_match_type	This field identifies which types of mix and match record this item belongs to. The item can either be a buy (exists on PROM_MIX_MATCH_BUY) or a get (exists on PROM_MIX_MATCH_GET) item. This field is only populated when the MIX_MATCH_NO is populated. Valid values are: B - Buy G - Get
	Threshold Number	Number(10)	Threshold_no	This field contains the number of the threshold in a promotion for which the discount originated. This field, along with the promotion, is used to isolate a list of items that tie together with discount information.

Record Name	Field Name	Field Type	Default Value	Description
	Launch Date	Char(8)	Launch_date	Date that the item should first be sold at this location, formatted to 'YYYYMMDD'.
	Quantity Key Options	Char(6)	Qty_key_options	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited.
	Manual Price Entry	Char(6)	Manual_price_entry	Determines whether the price can/should be entered manually on a POS for this item at the location. Valid values are in the code_type 'RPO'. Current values include 'R - required', 'P - Prohibited', and 'O - Optional'.
	Deposit Code	Char(6)	Deposit_code	Indicates whether a deposit is associated with this item at the location. Valid values are in the code_type 'DEPO'. Additional values may be added or removed as needed. Deposits are not subtracted from the retail of an item uploaded to RMS, etc. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Food Stamp Indicator	Char(1)	Food_stamp_ind	Indicates whether the item is approved for food stamps at the location.
	WIC Indicator	Char(1)	Wic_ind	Indicates whether the item is approved for WIC at the location.

Record Name	Field Name	Field Type	Default Value	Description
	Proportional Tare Percent	Number(12)	Proportional_tare_pct	Holds the value associated of the packaging in items sold by weight at the location. The proportional tare is the proportion of the total weight of a unit of an item that is packaging (i.e. if the tare item is bulk candy, this is the proportional of the total weight of one piece of candy that is the candy wrapper). The only processing RMS does involving the proportional tare percent is downloading it to the POS.
	Fixed Tare Value	Number(12)	Fixed_tare_value	Holds the value associated of the packaging in items sold by weight at the location. Fixed tare is the tare of the packaging used to (i.e. if the tare item is bulk candy, this is weight of the bag and twist tie). The only processing RMS does involving the fixed tare value is downloading it to the POS. Fixed tare is not subtracted from items sold by weight when sales are uploaded to RMS, etc. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Fixed Tare UOM	Char(4)	Fixed_tare_uom	Holds the unit of measure value associated with the tare value. The only processing RMS does involving the proportional tare value and UOM is downloading it to the POS. This kind of processing is the responsibility of the retailer and should occur before sales are sent to any Retek application.
	Reward Eligible Indicator	Char(1)	Reward_eligible_ind	Holds whether the item is legally valid for various types of bonus point/award programs at the location.

Record Name	Field Name	Field Type	Default Value	Description
	Elective Marketing Clubs	Char(6)	Elect_mtk_clubs	Holds the code that represents the marketing clubs to which the item belongs at the location. Valid values can belong to the code_type 'MTKC'. Additional values can be added or removed from the code type as needed
	Return Policy	Char(6)	Return_pocily	Holds the return policy for the item at the location. Valid values for this field belong to the code_type 'RETP'.
	Stop Sale Indicator	Char(1)	Stop_sale_ind	Indicates that sale of the item should be stopped immediately at the location (i.e. in case of recall etc).
	Returnable Indicator	Char(1)	Returnable_ind	Indicates that the item is returnable at the location when equal to ‘Y’es. Indicates that the item is not returnable at the location when equal to ‘N’o.
	Refundable Indicator	Char(1)	Refundable_ind	Indicates that the item is refundable at the location when equal to ‘Y’es. Indicates that the item is not refundable at the location when equal to ‘N’o.
	Back Order Indicator	Char(1)	Back_order_ind	Indicates that the item is back orderable at the location when equal to ‘Y’. Indicates that the item is not back orderable when equal to ‘N’o.
	Vat Code	Char(6)		Indicates the VAT code used with this item.
	Vat Rate	Number(20,10)		Indicates the VAT rate associated with this item and VAT code.
	Class Vat Indicator	Char(1)		Indicates whether or not the class VAT indicator is on or off for the class that this item exists in.
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type

Record Name	Field Name	Field Type	Default Value	Description
	File Line Identifier	Number ID(10)	Sequential number. Created by program.	ID of current line being created for output file.
	File Record Counter	Number ID(10)	Number of FDETL records. Created by program.	Number of records/transactions processed in current file (only records between head & tail)

Flat file used in the DexnexFileParser batch process

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 follows:

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 (e.g. 10% off) or charge (e.g. 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.

File details:

Segment	Sub-Segment	Name	Req?	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

Segment	Sub-Segment	Name	Req?	SIM value
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
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

Segment	Sub-Segment	Name	Req?	SIM value
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