Oracle® Retail Allocation

Operations Guide

Release 19.0

F26040-01

January 2020

ORACLE®

Oracle® Retail Allocation Cloud Service Operations Guide, Release 19.0

F26040-01

Primary Author: Nathan Young

reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

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

Contents

4 Batch Processes

Send Us Your Comments

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

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

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

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

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

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

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

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

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

Preface

The *Oracle Retail Allocation Operations Guide* provides critical information about the processing and operating details of the application, including the following:

- System configuration settings
- Technical architecture
- Functional integration dataflow across the enterprise
- Batch processing

Audience

This guide is for:

- Systems administration and operations personnel
- Systems analysts
- Integrators and implementation personnel
- Business analysts who need information about product processes and interfaces

Documentation Accessibility

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

Access to Oracle Support

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

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

- Product version and program/module name

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

Improved Process for Oracle Retail Documentation Corrections

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

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

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

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

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

Oracle Retail Documentation on the Oracle Technology Network

Oracle Retail product documentation is available on the following web site:

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

(Data Model documents are not available through Oracle Technology Network. You can obtain these documents through My Oracle Support.)

Conventions

The following text conventions are used in this document:

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

1
Introduction

Welcome to the Oracle Retail Allocation Operations Guide. The guide is designed so that you can view and understand key system-administered functions, the flow of data into and out of the application, and the application's behind-the-scenes processing of data.

Note: Users should not access Oracle Retail Allocation during the Merchandising batch window as it may cause some unpredictable results.

Allocation Overview

A retailer that acquires Oracle Retail Allocation gains the ability to achieve more accurate allocations on a stable product. Having the right product in the right stores or warehouses allows for service levels to be raised, sales to be increased, and inventory costs to be lowered. By accurately determining which locations should get which product, retailers can meet their turnover goals and increase profitability.

The Oracle Retail Allocation retailer benefits from the following capabilities:

- Built on ADF Technology stack, it allows the ability to quickly add UI based on ready to use design patterns, metadata driven tools and visual tools. Debugging can be performed more rapidly; maintenance and alteration costs are kept low using the metadata driven application development.
- The application's interface takes advantage of the Java Database Connectivity (JDBC), ADF's built-in transaction management, along with connections to data sources handled in WebLogic server which minimizes the interface points that need to be maintained.
- The application's robust algorithm executes rapidly, and the call to the calculation engine has been ported over from C++ library to a Java Library, thus minimizing the overhead/issues related to maintaining codebase consisting of two disparate languages.
- For retailers with other Oracle Retail products, integration with the Oracle Retail product suite means that item, purchase order, supplier, sales, and other data are accessed directly from the Merchandising tables, with no need for batch modules. Purchase order, item, location, and allocation information are passed from Merchandising to a warehouse management system.
- Access control to the system is managed by using Fusion Security Architecture.
- The application allows for retailers to adjust to changing trends in the market by facilitating real time allocations.

- Oracle Retail Allocation accounts for flexible supply chain paths to support importing and domestic inventory supply.

Merchandising owns virtually all of the information that Oracle Retail Allocation needs to operate, and the information that Oracle Retail Allocation provides is of primary interest/use for Merchandising. As a result, Oracle Retail Allocation has limited interaction with other Oracle Retail Merchandising Operations Management applications. For Oracle Retail Merchandising Operations Management applications that Oracle Retail Allocation does interact with, it is managed through direct reads from Oracle Retail Merchandising Operations Management application tables, direct calls to Oracle Retail Merchandising Operations Management packages, and Oracle Retail Allocation packages based on Oracle Retail Merchandising Operations Management application tables.

2

Integration

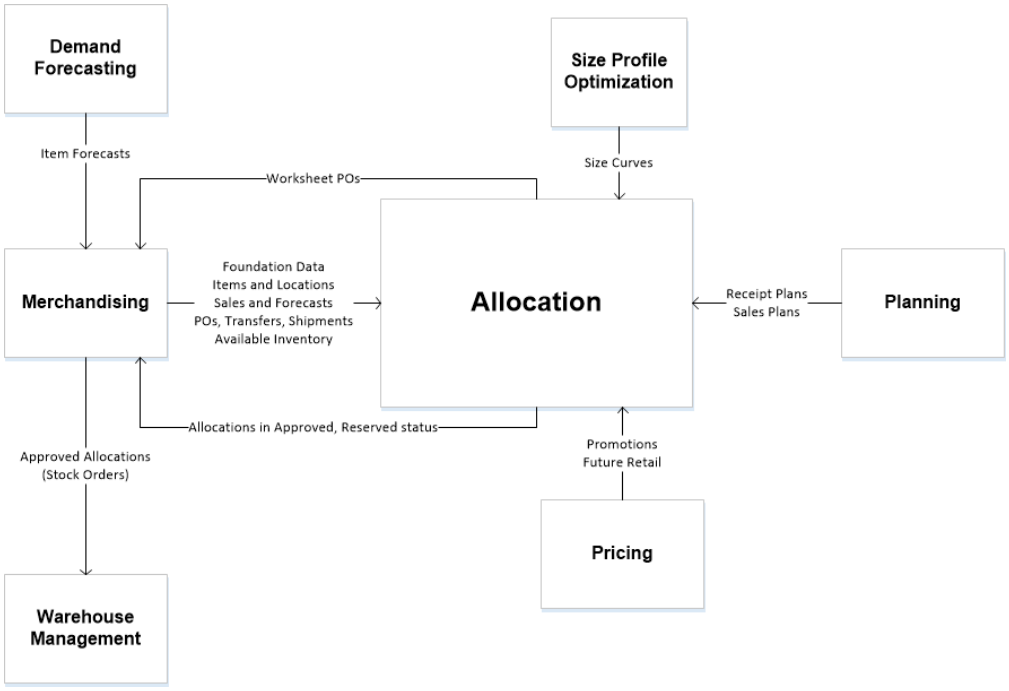
This chapter discusses the integration among Oracle Retail Allocation and other systems and it provides the following:

- An integration interface allocation-related dataflow across the enterprise.
- The tables and triggers that are in external systems or related to external systems that Allocation uses (for example, Merchandising).
- A functional description of Merchandising dependencies and assumptions that affect Oracle Retail Allocation.
- Information necessary to integrate Oracle Retail Allocation and Oracle Retail Workspace.

Allocation Integration Overview

This section provides an overview as to how Oracle Retail Allocation is functionally integrated with other systems (including other Oracle Retail systems). The discussion primarily concerns the flow of allocation-related business data across the enterprise.

Figure 2–1 Oracle Retail Allocation-Related Dataflow



Note: Oracle Retail Allocation pulls the data from the Merchandising tables using the JDBC connection.

The diagram above shows the overall direction of the dataflow among the products. The accompanying explanations are written from a system-to-system perspective, illustrating the movement of data.

From Oracle Retail Demand Forecasting

Oracle Retail Allocation pulls the following data from Merchandising:

- Forecasting data: Oracle Retail Allocation accesses forecasting data that originates in the Oracle Retail Demand Forecasting (RDF) system. RDF is Oracle Retail's statistical and causal forecasting solution. It uses state-of-the-art modeling techniques to produce high quality forecasts with minimal human intervention. RDF is an application that resides on the Oracle Retail Predictive Application Server (RPAS). Oracle Retail Allocation uses forecasting data as a basis for calculating gross need and can access the following five levels of forecasting data: department, class, subclass, style-color and item.

From Planning

Oracle Retail Allocation accesses plan data that originates in planning applications (including Oracle Retail's planning applications). The Oracle Retail Planning products are applications that provide functionality for developing, reconciling, and approving plans. When interfacing with Oracle Retail planning applications, Oracle Retail Allocation accesses department, class, subclass, parent/diff, or SKU plan data at the store-week level. Oracle Retail Allocation can be used as a tool to verify the final product-store plans and to initiate a PO to execute the plan. In other words, Oracle Retail Allocation can take the retailer's plan and execute it. Both the Oracle Retail and

the legacy planning applications populate a planning table, ALC_PLAN, which resides within Oracle Retail Allocation. See the section, "Planning Table in Oracle Retail Allocation," later in this chapter.

Note:

- Oracle Retail Allocation interfaces with Oracle Retail Assortment Planning by way of an output file from Assortment Planning through Oracle Retail Extract, Transform, and Load (RETL) to access only SKU, style-color data at the store-week level. For more information, see the chapter, "Oracle Retail Extract, Transform, and Load (RETL) Batch Processing".
- Merchandising is the system of record for Oracle Retail Allocation. Hence Allocation inherits the merchandise hierarchy from Merchandising. Merchandising and Assortment Planning (AP) have different merchandise hierarchies. Users of Merchandising/AP/Allocation who wish to export information from AP to Allocation must ensure that the AP merchandise hierarchy is compatible with that of Merchandising/Allocation.

From Size Profiles

Oracle Retail Allocation uses size profile data from Oracle Retail Size Profile Optimization (SPO) system or another similar size profile solution. SPO creates optimal profiles of size distribution by both merchandise category and by store. The size profile data is extracted at the following levels: department level, class level, sub-class level and item level. The data for all the levels are extracted in a single file. For more information, see the Oracle Retail Extract, Transform, and Load (RETL) Batch Processing chapter.

From Warehouse Management System

Allocation does not integrate with a WMS directly. Instead, it leverages its integration with Merchandising to have access to the following data:

- Appointments

  Appointment data is one source that identifies item(s) to be allocated.
- Warehouse inventory
- ASNs, BOLs, and Transfers

From Oracle Retail Pricing

Pricing provides the following to Allocation:

- **Future Retail Price Data** - Oracle Retail Allocation has the ability to get a real time price from Pricing as it is integrated directly with Pricing. Allocation uses this data to provide you with the future retail price value of the entire allocation (based on its quantities). In addition, you can access future retail price values by location and by item.

From Merchandising

Note: Item, purchase order, supplier, sales and other data are accessed directly from the Merchandising tables, with no need to interface data via batch modules.

- Items
- Purchase Orders
- Hierarchy data
- Sales history data (for items, user-defined attributes (UDA), warehouses, stores, and so on)
- Foundation data (supplier data, shipping tables, and so on)

From Allocation to Merchandising

Oracle Retail Allocation calculates the allocation based on the information it has received from Merchandising. Once the retailer reviews and approves the allocation, Oracle Retail Allocation sends the following information back to Merchandising:

- Approved or reserved allocation data
- Worksheet status POs that contain product, supplier and quantity information (the only remaining actions to be taken in Merchandising are to approve the PO and, if desired, to truck scale the PO.) These worksheet status purchase orders may be created or updated from within the Oracle Retail Allocation front end.

Based upon the approved allocation information from Oracle Retail Allocation, Merchandising sends the following information to the warehouse management system:

- Approved allocation data represents the store quantity instructions for allocating a specific quantity of stock at the store level.

Persistence Layer Integration

This section addresses Oracle Retail Allocation's persistence layer method of integration:

Persistence Layer Integration (Including Tables and Triggers)

Tables Populated by External Systems

The following tables are owned by Oracle Retail Allocation. The data within them is populated by external systems. For descriptions of each table and its columns, see the *Oracle Retail Allocation Data Model*.

- ALC_CORPORATE_RULE_DETAIL
- ALC_CORPORATE_RULE_HEAD
- ALC_IDEAL_WEEKS_OF_SUPPLY
- ALC_PLAN
- ALC_RECEIPT_PLAN

- ALC_SIZE_PROFILE
  - Can also be populated through size profile setup via the front end of the application.

Planning Table in Oracle Retail Allocation

Planning applications populate a planning table, ALC_PLAN, that resides within Oracle Retail Allocation. This table includes the following columns:

- Plan ID
- Location
- EOW date
- Department
- Class
- Subclass
- Item
- Diff1_id, Diff2_id, Diff3_id, Diff4_id
- Quantity

A record can thus exist at any of the following levels by week-store-quantity:

- Department
- Department-class
- Department-class-subclass
- Item-color
- Item

The ALC_RECEIPT_PLAN table includes the following columns:

- Plan ID
- Loc
- EOW.date
- Department
- Class
- Subclass
- Item
- Diff1_id, Diff2_id, Diff3_id, Diff4_id
- Quantity

A record can thus exist at any of the following levels by week-store-quantity:

- Department
- Department-class
- Department-class-subclass
- Item-color
- Item

- Pack

Merchandising Interface Tables

Oracle Retail Allocation and Merchandising share certain database tables and processing logic. This integration provides the following two important benefits:

- The number of interface points that need to be maintained is minimized.
- The amount of redundant data (required if the rest of the Oracle Retail product suite is installed) is limited.

Oracle Retail Allocation exchanges data and processing with Merchandising in four ways:

- By reading directly from Merchandising tables.
- By directly calling Merchandising packages.
- By reading Oracle Retail Allocation views based on Merchandising tables.
- Oracle Retail Allocation triggers reside in Merchandising tables. These triggers cause actions (create, delete, update) on Merchandising tables based on Oracle Retail Allocation business rules.

Merchandising Tables used by Allocation

The following table illustrates the tables from which Oracle Retail Allocation gets its data from Merchandising.

Table 2–1 Merchandising Tables Used by Allocation

Merchandising Tables	
Functional Area	Associated Tables
Item data	SUB_ITEMS_HEAD
	SUB_ITEMS_DETAIL
	ITEM_MASTER
	ITEM_SUPP_COUNTRY
	ITEM_SUPPLIER
	ITEM_LOC
	ITEM_LOC_HIST
	ITEM_LOC_SOH
	ITEM_PARENT_LOC_HIST
Skulist data	SKULIST_HEAD
	SKULIST_DETAIL
Pack data	PACKITEM
	ITEM_MASTER
	ITEM_LOC

Table 2–1 (Cont.) Merchandising Tables Used by Allocation

Merchandising Tables	
Order data	ORDHEAD
	ORDLOC_WKSHT
	ORDLOC
	ORDSKU
	ALLOC_HEADER
	ALLOC_DETAIL
	SHIPMENT
Supplier data	SUPS
	ITEM_SUPPLIER
Location list data	LOC_LIST_HEAD
	LOC_LIST_DETAIL
	LOC_LIST_CRITERIA
Merchandise hierarchy data	DEPS
	CLASS
	SUBCLASS
	ITEM_PARENT
	DIFF
	SKU
Organizational hierarchy data	STORE
	WH
	WH_STORE_ASSIGN
Shipment data	SHIPMENT
	SHIPSKU
Store grade data	STORE_GRADE_GROUP
	STORE_GRADE
	STORE
	BUYER
	STORE_GRADE_STORE
Location traits data	LOC_TRAITS
	LOC_TRAITS_MATRIX
Transfer data	TSFHEAD
	TSFDETAIL
User defined attribute (UDA) data	UDA
	UDA_VALUES
	UDA_ITEM_LOV

Table 2–1 (Cont.) Merchandising Tables Used by Allocation

Merchandising Tables	
Forecast data	DEPT_SALES_FORECAST
	CLASS_SALES_FORECAST
	SUBCLASS_SALES_FORECAST
	ITEM_FORECAST
Sales data	DEPT_SALES_HIST
	CLASS_SALES_HIST
	SUBCLASS_SALES_HIST
	ITEM_LOC_HIST
	ITEM_PARENT_LOC_HIST
Appointment data	APPT_HEAD
	APPT_DETAIL

Oracle Retail Allocation-Owned Triggers Residing on Merchandising Tables

Table 2–2 Triggers

Triggers	Details
ALC_TABLE_ALD_AUR - 1 - 4	This trigger is involved in the following processing: Whenever a portion of an allocation order is worked on by the distribution center by selecting, distributing, transferring or receiving inventory, the allocation within Oracle Retail Allocation is placed into a 'Processed' status. The user can no longer change that allocation in Oracle Retail Allocation.
ALLOC_STATUS_TRIGGER	This trigger is on the Merchandising table ALLOC_HEADER. The trigger updates the status in Oracle Retail Allocation table ALC_ALLOC to 4 (closed). This trigger is fired only if the status on Merchandising table ALLOC_HEADER is updated to 'C' (closed).
ALLOC_STATUS_ TRIGGER_AU	The closure logic within the Oracle Retail Allocation application accounts for the multiplicity between ALLOC_HEADER records and the Oracle Retail Allocation (ALC_XXX) tables. The table triggers only set a Oracle Retail Allocation allocation number to closed if all ALLOC_HEADER records have been closed.

3

Oracle Retail Extract, Transform, and Load (RETL) Batch Processing

The module works in conjunction with the Oracle Retail Extract Transform and Load (RETL) framework. This architecture optimizes a high performance data processing tool that allows database batch processes to take advantage of parallel processing capabilities.

The RETL framework runs and parses through the valid operators composed in XML scripts.

This chapter provides an overview of Oracle Retail Allocation RETL processing and defines the export file that is used when exporting Plan values. For more information on RETL, see the product's Programmer's Guide.

Note: In this chapter, some examples refer to RETL programs that are not related to Oracle Retail Allocation. References to these programs are included for illustration purposes only.

Note: The RETL loads into Allocation are point to point integration between Oracle Retail product, and are not designed to support generic uploads from other systems.

Functional Overview

The extracts from RETL may contain up to four levels of plan/profile. They consist of the department level, class level, subclass level and item level. All of these levels are contained in a single normalized file. Each record in the file has a dedicated 'space' and distinct position for department, class, subclass, item, store, diff1, diff2, diff3, diff4 and size profile quantity values and each record in the Plan file has a dedicated 'space' and distinct position for department, class, subclass, item, store, diff1, diff2, diff3, diff4, EOW date and plan quantity values. It is crucial that the records are mapped using the correct positions and space/padding rules for each data value.

- Regardless of the level of financial plan/profile, each record must include a store, diff value in one of the four diff value fields and a quantity value (including an EOW date for Plan only).
- Department-level financial plan or profiles include a department data value in the dedicated department field. The class, subclass and item fields do not contain any values. They remain empty.

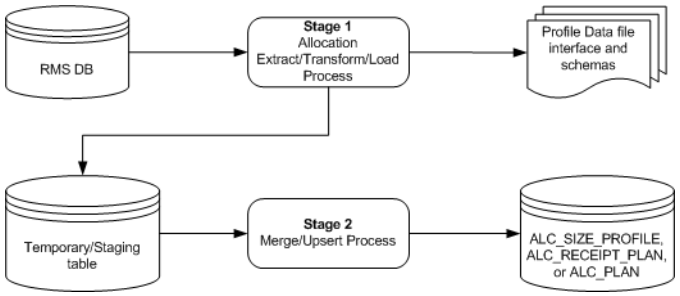
- Class-level financial plan or profiles include a department and class data value in the dedicated department and class fields. The subclass and item fields do not contain any values. They remain empty.
- Subclass-level financial plan or profiles include a department, class and subclass data value in the dedicated department, class and subclass fields. The item fields do not contain any values. They remain empty.
- All of the department, class and subclass record exports contain only the non-aggregate diff values mapped from the specific diff value in ITEM_MASTER to the corresponding diff value in the export file. It is crucial that the non-aggregate diffs are mapped to the correct diff_id in the export file.
- Item-level profiles include aggregate level IDs in the dedicated item field. The department, class and subclass fields do not contain any values. They remain empty. The item level export records contains both the aggregate and non-aggregate diff values mapped from the specific diff id in ITEM_MASTER to the associated diff position in the export file.
- Item-Level financial plan or profiles include item data value in the dedicated item field. The department, class and subclass fields do not contain any values. They remain empty.
- All the data values must start in the beginning of the corresponding field, and padding comes after the data to fill all the dedicated space for that data field.

Oracle Retail Extract, Transform, and Load Batch Processing Architecture

The diagram below illustrates the extraction processing architecture. The plan/size profile architecture adheres to what is shown in the diagram:

The architecture relies upon two distinct stages, each of which is described in the passages that follow.

Figure 3–1 RETL Batch Processing Architecture



Processing Stage 1

Stage 1 involves importing plan/profile data and looking up required information in the Merchandising ITEM_MASTER table (item-level plans/profiles only). The resulting output from this stage is a temporary table that contains any item-level and department/class/subclass-level plans/profiles.

The detailed flow is as follows:

1. Insert dept-level plans/profiles directly into the staging table.
2. Insert class-level plans/profiles directly into the staging table.
3. Insert subclass-level plans/profiles directly into the staging table.

4. The item-level plans/profiles require lookups with the ITEM_MASTER table. The processing logic for transaction-level items is to do a three-way left outerjoin on the ITEM_MASTER table to retrieve each parent and grandparent item aggregate indicators. The Item file then lookups its item id in the item master table and uses the STYLE set as the parent or grandparent item id whose ITEM_AGGREGATE_IND = 'Y'. These item level plans/profiles are then inserted into the staging table.

Error Handling

Any item records that do not have a parent and grandparent are flagged as warnings. Any items in the incoming data file that do not match an item in the ITEM_MASTER table are flagged as errors.

Processing Stage 2

Stage 2 involves inserting and updating the plan or profile records into the final destination ALC_PLAN, ALC_RECEIPT_PLAN, or ALC_SIZE_PROFILE table respectively.

The detailed processing is as follows:

1. Update quantity when matched department, class, subclass, style, store, size1, size2, size3, size4.
2. Otherwise, insert record.

Configuration

This section covers configuration.

Oracle Retail Extract, Transform, and Load

Before trying to configure and run Oracle Retail Allocation RETL, install RETL version 13.2, which is required to run Oracle Retail Allocation RETL. Run the 'verify_retl' script (included as part of the RETL installation) to ensure that RETL is working properly before proceeding.

Oracle Retail Extract, Transform, and Load User and Permissions

Oracle Retail Allocation RETL should be installed and run as the RETL user.

Additionally, the permissions should be set up as per the *RETL Programmer's Guide*. Oracle Retail Allocation RETL reads, creates, deletes and updates data for tables. If these permissions are not set up properly, processing fails.

Environment Variables

See the *RETL Programmer's Guide* for RETL environment variables that must be set up for your version of RETL. You need to set ALCHOME to your base directory for Oracle Retail Allocation RETL. This is the top level directory that you selected during the installation process (see Oracle Retail Allocation Installation Guide) in your .kshrc, you should add a line such as the following:

```
export ALCHOME=<base directory path for  ALLOCATION RETL>
```

Execute the setup-security-credential.sh script after the installation from RFX_HOME/bin directory. This script provides the options for adding/updating the database credentials. Enter the values for the following parameters:

- dbuseralias

- username

A secure wallet file (cwallet.sso) is created under "RFX_HOME/etc/security" directory of RETL installation.

alc_config.env Settings

On the Oracle Retail Allocation side, make sure to review the environmental parameters in the alc_config.env file before executing the batch module. Depending upon your local settings, the variables may need to be changed.

Configure Oracle Retail Extract, Transform and Load

1. Log in to the UNIX server with a UNIX account that runs the RETL scripts.
2. Change directory to $ALCHOME/rfx/etc.
3. Modify the alc_config.env script:
   - Change the DBNAME variable to the name of the Oracle Retail Allocation database. For example:

     ```
     export DBNAME=int9i
     ```

   - Set the user alias fields such as RETL_WALLET_ALIAS and ORACLE_WALLET_ALIAS.
   - Set the other variables namely: RFX_HOME, RFX_TMP, TNS_ADMIN, ALCHOME, ORACLE_HOME, JAVA_HOME, PATH, LD_LIBRARY_PATH, SPO_GID_FILENAME.

Update the rfx.conf, vdate.txt files with the DB information and other local settings necessary.

Running the Module

This section covers running the module.

Schema File

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

The input data schema file for the Oracle Retail Allocation module is named as alcl_plan.schema for Plan and as alcl_size_profile.schema for profile and is shown later in this chapter.

Mandatory Multi-Threading and Command Line Parameters

In contrast to the way in which multi-threading is defined in UNIX, Oracle Retail Allocation uses 'multi-threading' to refer to the running of a single RETL program multiple times on separate groups of data simultaneously. Multi-threading can reduce the total amount of processing time.

For this Oracle Retail Allocation module, multi-threading is *mandatory*, and the file-based module has to be run once for each input file.

- The alcl_pan / alcl_size_profile module requires the following two input parameters:
- The uniquely named thread number. Note that the thread number is used internally and is not related to any output file or table name.
- The following example illustrates a scenario in which the retailer runs the alcl_size_profile.ksh module three times for three input files:

```
The load batches, example alcl_size_profile.ksh, loads its data from $ALC_
HOME/data
alcl_size_profile.ksh profile_01.dat 1
alcl_size_profile.ksh profile_03.dat 3
```

The transform batches alct_plan and alct_size_profile do not take any input parameters. They execute the flat files in the path `$ALC_HOME/data alcl_size_profile.ksh profile_02.dat 2`. Running only `alct_size_profile.ksh` at the prompt executes the transform scripts.

Program Features

The extraction programs are written in the RETL framework and include the following features:

- Business virtual date
- Program return code
- Program status control files
- Restart and recovery
- Message logging
- Program error file
- Reject files

Business Virtual Date

The business virtual date must be placed in the vdate.txt file in the $ALCHOME/rfx/etc directory prior to running the RETL module.

Program Return Code

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

Program Status Control Files

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

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

File Naming Conventions

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

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

- The program name
- The input filename
- 'status'
- The business virtual date for which the module was run

For example, the program status control file for the alcl_size_profile.ksh program (with an input file name of alcl_size_profile_01.dat specified as the first argument on the command line) would be named as follows for the batch run of January 5, 2001:

```
$ALCHOME/error/alcl_size_profile.alcl_size_profile_01.dat.status.20010105
```

Oracle Retail Allocation Oracle Retail Extract, Transform, and Load Restart and Recovery

The Oracle Retail Allocation RETL module imports data from a flat file, performs transformations if necessary and then loads the data into the applicable Oracle Retail Allocation table.

This module uses a single RETL flow and does not require the use of restart and recovery. If the extraction process fails for any reason, the problem can be fixed, and the entire process can be run from the beginning without the loss of data. For a module that takes a text file as its input, the following two choices are available that enable the module to be re-run from the beginning:

1. Re-run the module with the entire input file.
2. Re-run the module with only the records that were not processed successfully the first time.

Message Logging

Message logs are written while the module is running in a format described in this section.

Daily Log File

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

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

- The business virtual date for which the module is run.
- '.log'

For example, the location and the name of the log file for the business virtual date of January 5, 2001 would be the following:

```
$ALCHOME/log/20010105.log
```

Format

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

```
alct_size_profile 16:06:30: Program started.Number of input files processed = 1.
Number of output files produced = 1
alcl_size_profile 13:20:01: Program started for thread 1…
alcl_size_profile 13:20:05: Analyzing temp table rmsint1201.alcl_size_profile_
temp_1
alcl_size_profile 13:20:13: Merging into alc_size_profile
alcl_size_profile 13:20:27: Program completed successfully for thread 1.
alct_size_profile 16:06:30: Program completed without errors
```

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

Program Error File

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

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

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

- The program name
- The business virtual date for which the module was run

For example, all errors and detail log information for the alcl_size_profile program would be placed in the following file for the batch run of January 5, 2001:

```
$ALCHOME/error/alcl_size_profile.alcl_size_profile_01.dat.20010105
```

The error file for the transform batch contains the name of the files processed with exit status of each file. If any of the file has bad record like width of some field exceeding the maximum allowed, this error file shows the bad records also.

The naming convention for the error file of the transform batch is

Program name_err.business virtual date for which the scripts were run.

```
For example,
alct_size_profile_err.20061005
```

Oracle Retail Allocation Reject Files

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

Typical Run and Debugging Situations

The following examples illustrate typical run and debugging situations for each type of program. The file names referenced in the example below (log, error, and so on) assume that the module is run on the business virtual date of March 9, 2001.

```
Example for running Transform batch
Run alct_size_profile.ksh
```

1. Change the directory to $ALC_HOME/src
2. In the prompt enter,

$alct_size_profile.ksh

If the module runs successfully, the following results,

```
alct_size_profile.ksh: 16:37:45 Data transform (awk) starting. Input file:
/home/pachaia/RPAS12_Agal/data/d1clss.01
alct_size_profile.ksh: 16:37:45 Transform completed. File: d1clss.01
d1clss.01 processing completed. Exit status: 0
alct_size_profile.ksh: 16:37:45 Data transform (awk) starting. Input file:
/home/pachaia/RPAS12_Agal/data/d1itpt.01
alct_size_profile.ksh: 16:37:45 Transform completed. File: d1itpt.01
d1itpt.01 processing completed. Exit status: 0
alct_size_profile completed with 1 ERRORS: Thu Oct  5 16:37:45 CDT 2006
If the module does not run successfully, the following results,
```

****** STARTING alct_size_profile: Thu Oct 5 16:37:45 CDT 2006 ******

```
******  STARTING alct_size_profile: Thu Oct  5 16:37:45 CDT 2006  ******
alct_size_profile.ksh: 16:37:45 Transform completed. File: d1clss.01
d1clss.01 processing completed. Exit status: 0
alct_size_profile.ksh: 16:37:45 Data transform (awk) starting. Input file:
/home/pachaia/RPAS12_Agal/data/d1dept.01
 ERROR - too many DIFF IDs in current record of Diff Profile File.
```

alct_size_profile.ksh: 16:37:45 Data transform (awk) starting. Input file: /home/pachaia/RPAS12_Agal/data/d1clss.01

```
Diff Profile file name: d1dept.01
 Number of Diffs found:  5
 Maximum number allowed: 4
 DIFF ID string: <  _hh1dif_jh2dif_kh3dif_lh4dif _mh5dif       000000006>
Bad Record:
```

Record number 2

```
1414                    1000000001             _hh1dif_jh2dif_kh3dif_lh4dif_
mh5dif        000000006000
alct_size_profile.ksh: 16:37:45 Transform completed. File: d1dept.01
d1dept.01 processing completed. Exit status: 2
alct_size_profile.ksh: 16:37:45 Data transform (awk) starting. Input file:
/home/pachaia/RPAS12_Agal/data/d1itpt.01
alct_size_profile.ksh: 16:37:45 Transform completed. File: d1itpt.01
d1itpt.01 processing completed. Exit status: 0
alct_size_profile completed with 1 ERRORS: Thu Oct  5 16:37:45 CDT 2006
```

Example for Running Load Batch

Run alcl_size_profile.ksh:

1. Change directory to $ALCHOME/rfx/src.
2. At a UNIX prompt, enter:

```
alcl_size_profile.ksh <input datafile 1> <thread #>
...
```

If the module runs successfully, the following results:

- Log file: Today's log file, 20010309.log, contains the messages described above in the 'Format' passage of the 'Daily log file' section.
- Data: The ALC_SIZE_PROFILE table exists in the Oracle Retail Allocation database and contains the extract records.
- Error File: The program's error file, alcl_size_profile.20010309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no error messages.
- Program status control: The program status control file, alcl_size_ profile.status.20010309, does not exist.
- Reject File: No reject files exist.

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

- Log file: Today's log file, 20010309.log, may not contain the "Program completed successfully…" message.
- Data: The ALC_SIZE_PROFILE table exists in the Oracle Retail Allocation database but may not contain all the records from the profile file interface.
- Error file: The program's error file, alcl_size_profile.20010309, may contain an error message.
- Program status control: The program status control file, alcl_size_ profile.status.20010309, may exist.
- Reject file: The reject file, items_not_found.dat, may exist in the $ALCHOME/data directory.
- Bookmark file: The bookmark file, alcl_size_profile.bkm.20010309, does not exist because this module does not utilize restart and recovery.

Re-run the Module:

1. Determine and fix the problem causing the error.
2. Remove the program's status control file.
3. Remove the reject file (if it exists) from the $ALCHOME/data directory.
4. Change directory to $ALCHOME/rfx/src. At a UNIX prompt, enter:

```
% alcl_size_profile.ksh <input datafile 1> <thread #>
```

Oracle Retail Allocation Program Reference

This section serves as a reference to the Oracle Retail Allocation program.

By reviewing this section and the section, 'API flat file specification', the retailer should be able to track down to the table and column level, all the extraction data that flows into Oracle Retail Allocation.

Table 3–1 Extraction Data

Program Name	Tables /Files Extracted	Fields Extracted	Target File or Table	Target Field	Field Type	Field Length	NOTES
alcl_size_ profile.ksh	ITEM_ MASTER	item	alc_size_ profile	style	VAR CHAR2 (25)	25	
	profile file	dept		dept	NUMBER (4)	4	
		class		class	NUMBER (4)	4	
		subclass		subclass	NUMBER (4)	4	
		store		store	NUMBER (10)	10	
		size1		size1	VAR CHAR2 (10)	10	
		size2		size2	VAR CHAR2 (10)	10	
		size3		size3	VAR CHAR2 (10)	10	
		size4		size4	VAR CHAR2 (10)	10	
		qty		qty	NUMBER (12,4)	17	

Table 3–1 (Cont.) Extraction Data

Program Name	Tables /Files Extracted	Fields Extracted	Target File or Table	Target Field	Field Type	Field Length	NOTES
ALC_ PLAN.KSH	ITEM_ MASTER	item	ALC_ PLAN	style	VAR CHAR2 (25)	25	
		item_ parent					Use item_ parent as style if Item_ parent_ aggregate_ ind= 'Y'
		item_ grandpare nt					Use item_ grandpare nt as Style if item_ grandpare nt_ aggregate_ ind = 'Y'
		item_ aggregate_ ind					
		item_ parent_ aggregate_ ind					
		item_ grandpare nt_ aggregate_ ind					
	profile file	dept		dept	NUMBER (4)	4	
		class		class	NUMBER (4)	4	
		subclass		subclass	NUMBER (4)	4	
		store		store	NUMBER (10)	20	
		size1		size1	VAR CHAR2 (10)	48	
		size2		size2	VAR CHAR2 (10)	48	
		size3		size3	VAR CHAR2 (10)	48	

Table 3–1 (Cont.) Extraction Data

Program Name	Tables /Files Extracted	Fields Extracted	Target File or Table	Target Field	Field Type	Field Length	NOTES
		size4		size4	VAR CHAR2 (10)	48	
		qty		qty	NUMBER (12,4)	12	
ALC_ RECEIPT_ PLAN.KSH	ITEM_ MASTER	item	ALC_ PLAN	style	VAR CHAR2 (25)	25	
		item_ parent					Use item_ parent as style if Item_ parent_ aggregate_ ind= 'Y'
		item_ grand parent					Use item_ grand parent as Style if item_ grandpare nt_ aggregate_ ind = 'Y'
		item_ aggregate_ ind					
		item_ parent_ aggregate_ ind					
		item_ grandpare nt_ aggregate_ ind					
	profile file	dept		dept	NUMBER (4)	4	
		class		class	NUMBER (4)	4	
		subclass		subclass	NUMBER (4)	4	
		store		store	NUMBER (10)	20	
		size1		size1	VAR CHAR2 (10)	48	
		size2		size2	VAR CHAR2 (10)	48	

Table 3–1 (Cont.) Extraction Data

Program Name	Tables /Files Extracted	Fields Extracted	Target File or Table	Target Field	Field Type	Field Length	NOTES
		size3		size3	VAR CHAR2 (10)	48	
		size4		size4	VAR CHAR2 (10)	48	
		qty		qty	NUMBER (12,4)	12	

Application Programming Interface (API) Specification

This section describes the API specification.

File Layout

Plan Data File Layout

Plan data is required in Allocation for allocating items to the store based on plan requirements at the location. The data is extracted by the predictive application server and provided to the allocation interfaces as a flat file.

This segment describes the file format that Predictive Application System provides for uploading plan data into Merchandising database for Allocation.

File Name: p1*prodlevel.NN*

Example: p1scls.01 (subclass, domain 01)

- 1234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123

- 141410001000 1000000014 _CCOLOR31_SMEDIUM

- 20051225000000137500

Table 3–2 Plan Data File Layout

Field Name	Start Position	Width	Format	Content
Product ID	1	25 char	Alpha	141410001000
Location ID	26	20 char	Alpha	10000000014
Diff IDs	46	48 char	Alpha	_CCOLOR31_ SMEDIUM
EOW Date	94	8 char	Alpha	20051225
Quantity	102	12 char	Numeric	000000137500

- The file name should start with p1 followed by four characters for product level and the domain number. The four product level acceptable are:
  - - itpt - for item
  - - scls - for subclass
  - - clss - for class

  - - dept - for department

- The domain ID should be numeric.
- The Product ID, Location ID and Diff IDs fields are left justified and blank filled.
- The number of separate Diffs in the Diff IDs field is in the range: 0-4. The first character of each Diff is an "_" (underscore) and the second character is the Diff Type. No underscore characters are present in the Diff ID field other than the character that immediately precedes each separate Diff Type within the field. Each Diff in the Diff IDs field is lesser than 12 characters in length, including the leading underscore character and the Diff Type.
- The EOW Date field is in the format YYYYMMDD.
- The Quantity field is a right-justified, zero-padded numeric and the decimal point is omitted, but the quantity has a 4-digit decimal fraction part (e.g. 13.75 would appear in the record as 000000137500).
- Total length of each record is 113.
- When uploading data the system updates the quantity if the record exists for the hierarchy/location/Diff_id/EOW data combination or it appends the record into the tables.

Receipt Plan Data File Layout

Receipt plan data is required in Allocation for allocating items to the store based on receipt plan requirements at the location. The data is extracted by the predictive application server and provided to the allocation interfaces as a flat file.

This segment describes the file format that Predictive Application System provides for uploading receipt plan data into Merchandising database for Allocation.

File Name: p1*prodlevel.NN*

Example: p1scls.01 (subclass, domain 01)

- 12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123
- 141410001000 1000000014 _CCOLOR31_SMEDIUM
- 20051225000000137500

Table 3–3 Receipt Plan Data File Layout

Field Name	Start Position	Width	Format	Content
Product ID	1	25 char	Alpha	141410001000
Location ID	26	20 char	Alpha	10000000014
Diff IDs	46	48 char	Alpha	_CCOLOR31_ SMEDIUM
EOW Date	94	8 char	Alpha	20051225
Quantity	102	12 char	Numeric	000000137500

- The file name should start with p1 followed by four characters for product level and the domain number. The four product level acceptable are:
  - - itpt - for item
  - - scls - for subclass

  - - clss - for class
  - - dept - for department
- The domain ID should be numeric.
- The Product ID, Location ID and Diff IDs fields are left justified and blank filled.
- The number of separate Diffs in the Diff IDs field is in the range: 0-4. The first character of each Diff is an "_" (underscore) and the second character is the Diff Type. No underscore characters are present in the Diff ID field other than the character that immediately precedes each separate Diff Type within the field. Each Diff in the Diff IDs field is lesser than 12 characters in length, including the leading underscore character and the Diff Type.
- The EOW Date field is in the format YYYYMMDD.
- The Quantity field is a right-justified, zero-padded numeric and the decimal point is omitted, but the quantity has a 4-digit decimal fraction part (e.g. 13.75 would appear in the record as 000000137500).
- Total length of each record is 113.
- When uploading data the system updates the quantity if the record exists for the hierarchy/location/Diff_id/EOW data combination or it appends the record into the tables.

Size Curve File Layout

Allocation application allocates quantities at the style-color level for fashion merchandise like Blue t-shirts; Dark Blue pants etc and then distributes the allocated quantity according to the size based on the size curve for the location. The size curve data can be inputted into the application directly using the GUI interface in the application or through upload from data provided by the predictive application server.

This section describes in detail the file format structure for the upload.

File Name: d*X*_*prodlevel*_*.NN*

Example: File Name: d1scls.01 (diff 1, subclass, domain 01)

Record:

- 12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345
- 100045078 1000000002 CCOLOR01_S30x32 000000137500

Table 3–4 Size Curve File Layout

Field Name	Start Position	Width	Format	Content
Product ID	1	25 char	Alpha	100045078
Location ID	26	20 char	Alpha	1000000002
Diff IDs	46	48 char	Alpha	_ CCOLOR01_ S30x32
Quantity	94	12 char	Numeric	000000137500

- The file name should start with letter d, X is diff number being sent followed by four characters for product level and the domain number. The four product level acceptable are:
  - - itpt - for item
  - - scls - for subclass
  - - clss - for class
  - - dept - for department
- The domain ID should be numeric.
- The Product ID, Location ID and Diff IDs fields are left justified and blank filled.
- The number of separate Diffs in the Diff IDs field is in the range: 0-4. The first character of each Diff is an "_" (underscore) and the second character is the Diff Type. No underscore characters is present in the Diff ID field other than the character that immediately precedes each separate Diff Type within the field. Each Diff in the Diff IDs field is lesser than 12 characters in length, including the leading underscore character and the Diff Type.
- The Quantity field is a right-justified, zero-padded numeric and the decimal point is omitted, but the quantity has a 4-digit decimal fraction part (e.g. 13.75 would appear in the record as 000000137500).
- Total length of each record is 105.
- When uploading data the system updates the quantity if the record exists for the hierarchy/location/Diff_id/EOW data combination or it appends the record into the tables.

Schema file (alcl_size_profile.schema)

This section describes the RETL schema file (alcl_size_profile.schema) used in the RETL script that loads the export file into Allocation's ALC_SIZE_PROFILE table:

```
<RECORD type="fixed" len="115" final_delimiter="0x0A">
<!-- start pos 5  --> <FIELD name="CLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 9  --> <FIELD name="SUBCLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 13 --> <FIELD name="ITEM" len="25" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 38 --> <FIELD name="STORE" len="20" datatype="string"
nullable="false"/>
<!-- start pos 58 --> <FIELD name="DIFF_TYPE_1" len="1" datatype="string"
nullable="false" nullvalue=""/>
<!-- start pos 59 --> <FIELD name="DIFF1" len="10" datatype="string"
nullable="false" nullvalue=""/>
<!-- start pos 69 --> <FIELD name="DIFF_TYPE_2" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 70 --> <FIELD name="DIFF2" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 80 --> <FIELD name="DIFF_TYPE_3" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 81 --> <FIELD name="DIFF3" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 91 --> <FIELD name="DIFF_TYPE_4" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 92 --> <FIELD name="DIFF4" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 102 --> <FIELD name="QTY" len="14" datatype="dfloat"
```

```
nullable="false"/>
<!-- end pos 114  -->
</RECORD>
<!-- start pos 1 --> <FIELD name="DEPT" len="4" datatype="string" nullable="true"
nullvalue=""/>
```

Schema file (alcl_plan.schema)

This section describes the RETL schema file (alcl_plan.schema) used in the RETL script that loads the plan export file into Allocation's ALC_PLAN table.

```
<RECORD type="fixed" len="123" final_delimiter="0x0A">
<!-- start pos 5  --> <FIELD name="CLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 9  --> <FIELD name="SUBCLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 13 --> <FIELD name="ITEM_ID" len="25" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 38 --> <FIELD name="STORE" len="20" datatype="string"
nullable="false"/>
<!-- start pos 58 --> <FIELD name="DIFF_TYPE_1" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 59 --> <FIELD name="DIFF1_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 69 --> <FIELD name="DIFF_TYPE_2" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 70 --> <FIELD name="DIFF2_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 80 --> <FIELD name="DIFF_TYPE_3" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 81 --> <FIELD name="DIFF3_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 91 --> <FIELD name="DIFF_TYPE_4" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 92 --> <FIELD name="DIFF4_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 102 --> <FIELD name="EOW_DATE" len="8" datatype="date"
nullable="false"/>
<!-- start pos 110 --> <FIELD name="QTY" len="14" datatype="dfloat"
nullable="false"/>
<!-- end pos 122  -->
</RECORD>
<!-- start pos 1 --> <FIELD name="DEPT" len="4" datatype="string" nullable="true"
nullvalue=""/>
```

Schema File (alcl_receipt_plan.schema)

This section describes the RETL schema file (alcl_receipt_plan.schema) used in the RETL script that loads the receipt plan export file into Allocation's ALC_RECEIPT_ PLAN table:

```
<RECORD type="fixed" len="123" final_delimiter="0x0A">
<!-- start pos 5  --> <FIELD name="CLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 9  --> <FIELD name="SUBCLASS" len="4" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 13 --> <FIELD name="ITEM_ID" len="25" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 38 --> <FIELD name="STORE" len="20" datatype="string"
nullable="false"/>
<!-- start pos 58 --> <FIELD name="DIFF_TYPE_1" len="1" datatype="string"
nullable="true" nullvalue=""/>
```

```
<!-- start pos 59 --> <FIELD name="DIFF1_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 69 --> <FIELD name="DIFF_TYPE_2" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 70 --> <FIELD name="DIFF2_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 80 --> <FIELD name="DIFF_TYPE_3" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 81 --> <FIELD name="DIFF3_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 91 --> <FIELD name="DIFF_TYPE_4" len="1" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 92 --> <FIELD name="DIFF4_ID" len="10" datatype="string"
nullable="true" nullvalue=""/>
<!-- start pos 102 --> <FIELD name="EOW_DATE" len="8" datatype="date"
nullable="false"/>
<!-- start pos 110 --> <FIELD name="QTY" len="14" datatype="dfloat"
nullable="false"/>
<!-- end pos 122  -->
</RECORD>
<!-- start pos 1 --> <FIELD name="DEPT" len="4" datatype="string" nullable="true"
nullvalue=""/>
```

Oracle Retail Extract, Transform, and Load for Receipt and Plan

RETL scripts are required to support receipt plan logic (what the store is expected to own) at the store/week level. This rule provides fashion retailers the option to choose different plans coming from different source data feeds.

Large size fashion retailers can use Assortment Planning data as the pre-season source to determine store or warehouse needs for seasonal items. The data file sent from the Assortment Planning system is used to generate the gross need in the Allocation system in order to create pre-allocations.

Once the selling season begins, retailers have the option to switch to a different rule such as Plan, Forecast or Historical data to generate store demand.

Script Names

- alct_receipt_plan.ksh
- alct_receipl_plan.ksh

Table Name

- alc_receipt_plan

Oracle Retail Extract, Transform, and Load for Size Profile Optimization Data

Allocation users have the option to select a specified store size profile to be used for the Allocation. Using the RPAS Store Size Profile Optimization application, users have the capability to create seasonal store size profiles and multiple store size profiles created in SPO (called GIDs). These are displayed to the Allocation user as options to be used.

- Depending on what is being allocated and expected arrival date in the stores, the Allocation user has the option to view and select the desired store size profile date to be used.
- All item and locations use the same store size profile data per allocation. There cannot be unique buy item/location records within a single allocation.

- SPO assigns a numeric generation ID number (GID) to specifically created store size profile data. This ID, along with a user defined name should be displayed in the Allocation user interface.
- Only those GIDs populated from SPO to Allocation are displayed in the user interface.
- The retailer is responsible for updating the Allocation table on a frequent basis or as needed.

SPO GID text files (spo_gid_label.txt) are passed along with the batch of Size Profile Hierarchy dat file. The text file is used as the GID for that batch of dat file. Running RETL for SPO imports data to three tables after extraction.

The RETL for SPO data file format is as follows:

<Beginning of file>

<GID>

<GID_DESC>

<End of File>

The following are examples:

GID1

Winter 2014

Table 3–5 RETL for SPO Data Physical Tables

Table Head	Description
ALC_GID_HEADER	The ALC_GID_HEADER table holds all generation ID descriptions.
ALC_GID_PROFILE	The ALC_GID_PROFILE table holds all generation ID profile IDs.
ALC_SIZE_PROFILE	The ALC_SIZE_PROFILE table holds all size profiles at Style/Color, Style, Subclass, Class and Department levels.

Limitations of Oracle Retail Extract, Transform, and Load Programs

The three programs that exist for receipt plan and plan and size profile have the following limitations:

- The diff type is supported to a maximum of 1 character length.
- The diff id is supported to a maximum of 10 character length.

4

Batch Processes

This chapter provides an overview of the batch processes of Oracle Retail Allocation. It also provides information about the functions of the batch processes, the packages associated with the batches, and how to execute the Java-based batches.

Batch Processing Overview

Allocation contains a set of batch processes that are run in Java. Broadly, the batch processing falls into five categories:

- Schedule Allocation batch - ScheduledAllocationBatchClient.java creates the child allocations for parent allocations that are scheduled for the day.
- Daily Cleanup batch - SessionCleanUpBatchClient.java deletes data from the temporary tables used by the Allocations and Calculation engine.
- Purge batches - Purge batches delete Allocation and Worksheet data from the Allocation tables, which were created before a certain time period.
- Rule Level On Hand (RLOH) batches - For RLOH, there are six batch update processes that share the same java batch file; InventorySnapshotBatchClient.java.
- Dashboard Refresh batch - The Dashboard Refresh batch refreshes both Stock to Sales and Top to Bottom Dashboard reports Data.

Batch Processes

The following table describes Oracle Retail Allocation's batch processes:

Table 4–1 Allocation's Batch Process and associated Java Packages

Batch Name	Batch Process	Package
Schedule Allocation Batch	ScheduledAllocationBatchClient.java	oracle.retail.apps.alc.batch.client
Daily Cleanup Batch	SessionCleanUpBatchClient.java	oracle.retail.apps.alc.batch.client
Purge Batches	PurgeBatchRunnable.java	oracle.retail.apps.alc.batch.client
RLOH Batch Update	InventorySnapshotBatchClient.java	oracle.retail.apps.alc.batch.client
Dashboard Refresh Batch	AlcDashboardCleanUp.ksh	oracle.retail.apps.alc.batch.client

Running a Java-based Batch Process

To run a Java-based batch process, Oracle Retail provides sample shell scripts (.sh files) and batch files (.bat files). These sample shell scripts must be modified according to the retailer's installation. They perform the following internally:

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

Scheduler and 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.

For UNIX systems, the Java process is scheduled through an executable shell script (.sh file).

Note: The `AllocScheduleBatch.ksh` and `AlcDailyCleanUp.ksh` batches can be run by an external scheduling system or a simple UNIX CRON job.

Running the Dashboard Refresh Batch

Take the following steps to run the Daily Cleanup batch:

1. Login to the application server machine using <username>/<password>.
2. Navigate to the batch folder. In the batch folder, verify that the AlcDashboardCleanUp.ksh file is present.
3. Run the AlcDashboardCleanUp.ksh batch using the following command:

```
ksh AlcDashboardCleanUp.ksh <systemadministratoralias>
```

The batch runs by taking the batch user from wallet.

Running the Schedule Allocation Batch

Installation and build scripts create the required user for running the batch in the wallet. There is no way you can cross check to determine whether the user is created inside the wallet other than running the batch scripts. However, you can see if the wallet is present in the environment by checking the wallet location. The wallet location is present in batch.properties file. The wallet is created with a user_id, password and an alias name.

Once the wallet is created the csm.wallet.path key in batch.properties file should be updated.

Only those users who have their role mapped to the SYSTEM_ADMINISTRATOR_JOB enterprise role in LDAP, have the privilege to execute the Schedule Allocation batch script. During installation, Allocation creates the SYSTEM_ADMINISTRATOR user, by default, in the Retail Wallet, which is mapped to the SYSTEM_ADMINISTRATOR_JOB enterprise role in LDAP. An alias for any new user mapped to SYSTEM_ADMINISTRATOR_JOB role in LDAP has to be created in the Wallet in order to execute the Schedule Allocation batch script.

Note: Use the `save_credential.sh` script to create a new user in the Wallet. For more information on instructions to run the `save_credential.sh script` to add a new user, see the *Oracle Retail Allocation Installation Guide*.

The batch.properties file exposes a few configuration parameters related to concurrency management or parallel execution which need to be tuned by the retailer based on the volume of transactions. The concurrent processing in batch is implemented leveraging the standard Java Executor service APIs. The sample file with default configurations will be made available and need to be modified by the retailer to suit to their requirements.

The section below describes the properties that can be configured.

- initialThreadLimit: Initial number of threads in the pool that are available to create child allocations. The default value is 5.
- maxThreadLimit: Maximum number of threads that can be allowed in the pool. The default value is 10.
- queueLimit: Size of queue of pending tasks to create child. The default value is 1.
- providerUrl: Url of the server module (for example, t3://<weblogic host>:<port>). This parameter has to be configured by the retailer to point to the WebLogic Server on which Asynchronous application instance is deployed.
- csm.wallet.partition.name: Partition name in the wallet (for example, alloc13)
- csm.wallet.path: Location of Wallet

1. Login to the application server machine using <username>/<password>.
2. Navigate to the batch folder. If the batch folder is not found, the batch installation did not occur properly. In the batch folder, verify that the `AllocScheduleBatch.ksh` file is present.
3. Run the `AllocScheduleBatch.ksh` batch using the following command:

   ```
   ksh AllocScheduleBatch.ksh <systemadministratoralias>
   ```

   The batch runs by taking the batch user from wallet.

Running the Daily Cleanup Batch

Take the following steps to run the Daily Cleanup batch:

1. Login to the application server machine using <username>/<password>.
2. Navigate to the batch folder. In the batch folder, verify that the `AlcDailyCleanUp.ksh` file is present.
3. Run the AlcDailyCleanUp.ksh batch using the following command:

   ```
   ksh AlcDailyCleanUp.ksh <systemadministratoralias>
   ```

   The batch runs by taking the batch user from wallet.

Running the Purge Batches

Use the following steps to run the Purge batches:

1. Login to the application server machine using <username>/<password>.

2. Navigate to the batch folder. In the batch folder, verify that the AlcPurgeAlloc.ksh and AlcPurgeWksht.ksh files are present.
3. Run the both batch processes using the following command:

```
ksh AlcPurgeAlloc.ksh <systemadministratoralias> PURGE_ALLOC
ksh AlcPurgeWksht.ksh <systemadministratoralias> PURGE_WORKSHEET
```

   The batch runs by taking the batch user from wallet.

Running the Rule Level On Hand Batch

Take the following steps to run the RLOH batch:

1. Login to the application server machine using <username>/<password>. Once logged in, the default folder is /home/alcbatch.
2. Before running the batch, make sure that all the corresponding profile properties are set. For that run the profile file first. Go to the **Profiles** folder inside alcbatch.
3. If there are multiple environments, there are separate profile files for every machine (for example, QA, DEV, TEST). Make sure to identify the right profile file here. Most likely it will be the name of the environment, run the profile file - ./alc132Linuxdev (for example, Dev 13.2 Env).
4. After running the profile successfully, go back to alcbatch. There are separate folders for every machine's batch under the **alcbatch** folder. Go to the current machine's folder. (Most likely the folder name would be same as your profile file name, in this case alc132Linuxdev).
5. Run the following scripts inside the batch folder in the following order:
   - ksh AlcSnapshotSOH.ksh <BatchUserAlias>
   - ksh AlcSnapshotOnOrder.ksh <BatchUserAlias>
   - ksh AlcSnapshotAllocIn.ksh <BatchUserAlias>
   - ksh AlcSnapshotCrosslink.ksh <BatchUserAlias>
   - ksh AlcSnapshotAllocOut.ksh <BatchUserAlias>
   - ksh AlcSnapshotCustomerOrder.ksh <BatchUserAlias>

Summary of Executable Files

The following table describes the executable shell scripts and batch files:

Table 4–2 Scripts to initiate the deletion process

Executable Shell Scripts (UNIX)	Executable Batch File For Windows	Description
AllocScheduleBatch.ksh	No batch file is available	Triggers the schedule batch client.
AllocBatch.ksh	No batch file is available	Configures the environment variables sourced by other batch scripts. This script is not to be run/scheduled in a stand alone mode.
AlcSnapshotSOH.ksh	No batch file is available	
AlcSnapshotOnOrder.ksh	No batch file is available	

Table 4–2 (Cont.) Scripts to initiate the deletion process

Executable Shell Scripts (UNIX)	Executable Batch File For Windows	Description
AlcSnapshotAllocOut.ksh	No batch file is available	
AlcSnapshotCustomerOrder.ksh	No batch file is available	
AlcSnapshotCrosslink.ksh	No batch file is available	
AlcSnapshotAllocIn.ksh	No batch file is available	
AlcDailyCleanUp.ksh	No batch file is available	Deletes data from the temporary tables.
AlcPurgeAlloc.ksh	No batch file is available	Deletes old Allocations from database table
AlcPurgeWksht.ksh	No batch file is available	Deletes old Worksheets from database table

AllocScheduleBatch Process Batch Design

The Allocation Auto Scheduler creates child allocations on pre-defined days of the week set by the Allocation user within the user interface. These allocations are created from an existing parent allocation. The auto creation of the child allocations must be called daily via a batch process at a scheduled time, set by the system administrator.

This process needs to be scheduled to run every day (using an external scheduling framework or UNIX CRON job).

Usage

The following command runs the AllocScheduleBatch job:

```
AllocScheduleBatch.ksh userAlias
```

Detail

This script is present under the $ALLOCHOME/batch folder.

Log File

log4j.xml is present under $ALLOCHOME/properties folder. This file is edited to specify desired log file location and name. To perform this action, change the value against param with name="file" in log4j.xml. Make sure that folder is already present on the file system and the batch user has write permission. Default value is set to ../logs/alloc133.log.

Properties File

The default batch properties file is present under $ALLOCHOME/properties/oracle/retail/alloc/batch.properties.

The properties below are defined. The default value may be edited.

- initialThreadLimit initial number of threads in the pool that are available to create child allocations. The default value is 5.

- maxThreadLimit maximum number of threads that can be allowed in the pool. The default value is 10.
- queueLimit size of queue of pending tasks to create child. The default value is 1.
- initialContextFactory specifies the JNDI context factory class (this should not be changed).
- providerUrl url of the server module (e.g t3://<weblogic host>:<port>). This parameter has to be configured by the retailer to point to the WebLogic Server on which the asynchronous application instance is deployed.
- csm.wallet.partition.name is the partition name in the wallet that stores the credentials to authenticate batch user on WebLogic. For example, alloc13
- csm.wallet.path is the path of the wallet file that stores WebLogic credentials.

Configuration

$ALLOCHOME/batch/AllocBatch.ksh should be edited by the retailer to specify appropriate value of following environment variables

- ALLOCHOME: directory where batch client in installed
- JAVA_HOME: directory where JDK is installed

Assumptions and Scheduling Notes

This job should be scheduled to run every day at the same time.

AlcDailyCleanUp Process Batch Design

Allocation has a number of temporary tables that store intermediate data while creating allocations and while performing calculations. The Daily Cleanup batch process deletes data from these temporary tables. Run this batch immediately after you run the Schedule Allocation batch.

This process should be scheduled to run every day (using an external scheduling framework or UNIX CRON job).

Make sure you run this process while all users are offline from the system.

Usage

The following command runs the AlcDailyCleanUp job:

```
AlcDailyCleanUp.ksh <BatchUserAlias>
```

Detail

This script is present under the $ALLOCHOME/batch folder.

The temporary tables which are impacted by the AlcDailyCleanUp process are as follows:

Allocation session tables:

- ALC_SESSION_SIZE_PROFILE_RATIO
- ALC_SESSION_SIZE_PROFILE
- ALC_SESSION_QUANTITY_LIMITS

- ALC_SESSION_ITEM_LOC_EXCL
- ALC_SESSION_ITEM_LOC
- ALC_SESSION_GID_PROFILE_LIST
- ALC_SESSION_GID_PROFILE

Worksheet session tables:

- ALC_WORK_SESSION_ITEM_LOC
- ALC_WORK_SESSION_ITEM_ALL
- ALC_WORK_SESSION_ITEM

Temporary tables:

- ALC_LOAD_TEMP

  alc_calc_destination_temp

  alc_calc_need_temp

  alc_calc_rloh_temp

  alc_calc_qty_limits_temp

  alc_calc_rloh_item_temp

  alc_merch_hier_rloh_temp

  alc_calc_source_temp

  alc_calc_need_dates_temp

Allocation approval tables:

- ALC_SYNC_HEADER_TEMP
- ALC_SYNC_DETAIL_TEMP

AlcPurgeAlloc AlcPurgeWksht Batch Processes Design

Allocation has a number of temporary tables that store intermediate data while creating allocations and while performing calculations. The Purge batch process deletes data from these temporary tables. Run this batch immediately after you run the Schedule Allocation batch. This process should be scheduled to run every day using an external scheduling framework. Make sure you run this process while all users are offline from the system.

Along with the above mentioned capability, this batch also allows provides for deletion of older allocations and worksheets created as a part of the Allocation application.

Usage

The following command runs the job:

```
AlcPurgeAlloc.ksh <systemadministratoralias> PURGE_ALLOC

AlcPurgeWksht.ksh <systemadministratoralias> PURGE_WORKSHEET
```

Details:

Allocation deletions are driven by the system option ALLOCATION_RETENTION_DAYS. Allocations exceeding the retention parameter become purge candidates as follows:

- Scheduled Allocations Parents are deleted when their scheduled end date is greater than the allocation retention days parameter.
- Allocations that are linked to Merchandising allocations in the ALC_XREF table are deleted when the Merchandising allocations they are linked to no longer exist in Merchandising.
- Allocations that are not linked to Merchandising allocations in the ALC_XREF table are deleted when they have not been modified (ALC_ALLOC.LAST_UPDATE_DATE) for ALC_SYSTEM_OPTIONS.TP_ALLOC_RETENTION_DAYS days.
- Allocations in Deleted status - user deleted through the UI.

Worksheets not associated to an allocation (WK worksheets) are deleted based on this setting. Worksheets associated to an allocation (WD worksheets) are deleted when the allocation they are related to is deleted (they follow Allocation deletion). Worksheet deletion is driven by a system option, WORKSHEET_RETENTION_DAYS. Worksheets purge criteria is as follows:

Worksheets not tied to an allocation (type = WK) are deleted when they are not be modified (ALC_WORK_HEADER.UPDATED_DATE) for TP_WORKSHEET_RETENTION_DAYS days.

Rule Level On Hand Pre-Aggregation Inventory Snapshot Batch Design

This batch process addresses the most significant performance issue within the Allocation product, the rule level on hand (RLOH) logic. This functionality requires current and future inventory lookups for potentially entire departments.

Inventory is currently only held in Merchandising at the transaction level item level. Departments in Merchandising can have tens of thousands of items under them. Multiply this by the hundreds of locations that can be on an allocation and RLOH can easily end up needing to retrieve inventory for millions of item/location combinations.

Usage

Six separate executables are called by one java batch process. The executables and the commands to run them are as follows:

- AlcSnapshotOnOrder.ksh

  ```
  ./AlcSnapshotOnOrder.ksh <BatchUserAlias>
  ```

- AlcSnapshotCrosslink.ksh

  ```
  ./AlcSnapshotCrosslink.ksh <BatchUserAlias>
  ```

- AlcSnapshotAllocIn.ksh

  ```
  ./AlcSnapshotAllocIn.ksh <BatchUserAlias>
  ```

- AlcSnapshotSOH.ksh

  ```
  ./AlcSnapshotSOH.ksh <BatchUserAlias>
  ```

- AlcSnapshotAllocOut.ksh

```
./AlcSnapshotAllocOut.ksh <BatchUserAlias>
```

- AlcSnapshotCustomerOrder.ksh

```
./AlcSnapshotCustomerOrder.ksh <BatchUserAlias>
```

A remote interface can be called for each batch

```
public interface IInventorySnapshotCoreRemote {

    public void createItemLocSOHSnapshot() throws AllocRemoteException;

    public void createOnOrderSnapshot() throws AllocRemoteException;

    public void createAllocInSnapshot() throws AllocRemoteException;

    public void createCrosslinkInSnapshot() throws AllocRemoteException;

    public void createAllocOutSnapshot() throws AllocRemoteException;

    public void createCustomerOrderSnapshot() throws AllocRemoteException;
}
```

Each method lines up with the appropriate PL-SQL function.

Detail

Retrieving inventory requires accessing four very large Merchandising tables:

- ITEM_LOC_SOH - current inventory and components of future inventory
- ORDLOC - on order component of future inventory
- ALLOC_DETAIL - allocation in component of future inventory
- TSFDETAIL - crosslink transfer component of future inventory

To improve RLOH performance, four new subclass level aggregated tables are created for use by Allocation

Table 4–3 RLOH Aggregated Tables

Table	Candidate Key	Source Tables
SUBCLASS_ITEM_LOC_SOH_EOD	Dept	ITEM_LOC_SOH
	Class	ITEM_MASTER
	Subclass	
	Loc	
SUBCLASS_ON_ORDER_EOD	Dept	ORDLOC
	Class	ORDHEAD
	Subclass	ITEM_MASTER
	Loc	PACKITEM_BREAKOUT
	On_order_date	

Table 4–3 (Cont.) RLOH Aggregated Tables

Table	Candidate Key	Source Tables
SUBCLASS_ALLOC_IN_EOD	Dept	ALLOC_DETAIL
	Class	ALLOC_HEADER
	Subclass	ITEM_MASTER
	Loc	PACKITEM_BREAKOUT
	Alloc_in_date	ORDHEAD
		TSFHEAD
SUBCLASS_CROSSLINK_EOD	Dept	TSFHEAD
	Class	TSFDETAIL
	Subclass	ITEM_MASTER
	Loc	PACKITEM_BREAKOUT
SUBCLASS_ALLOC_OUT_EOD	Dept	ALLOC_DETAIL
	Class	ALLOC_HEADER
	Subclass	ITEM_MASTER
	Loc	ORDHEAD
	Alloc_Out_Date	PACKITEM_BREAKOUT
ALC_SUBCLASS_CUST_ORDER_EOD	Dept	TSFHEAD
	Class	TSFDETAIL
	Subclass	ITEM_MASTER
	Loc	PACKITEM_BREAKOUT

These tables are populated by the ALC_HIER_LVL_INV_SNAPSHOT_SQL package (called by a batch program) nightly.

Package Details

The package that needs to be called is ALC_HIER_LVL_INV_SNAPSHOT_SQL.

```
SQL> desc ALC_HIER_LVL_INV_SNAPSHOT_SQL
FUNCTION ROLLUP_ALLOC_IN RETURNS NUMBER
Argument Name                  Type                      In/Out Default?
------------------------------ ------------------------ ------ --------
O_ERROR_MESSAGE                VARCHAR2                  IN/OUT
FUNCTION ROLLUP_CROSSLINK_IN RETURNS NUMBER
Argument Name                  Type                      In/Out Default?
------------------------------ ------------------------ ------ --------
O_ERROR_MESSAGE                VARCHAR2                  IN/OUT
FUNCTION ROLLUP_IL_SOH RETURNS NUMBER
Argument Name                  Type                      In/Out Default?
------------------------------ ------------------------ ------ --------
O_ERROR_MESSAGE                VARCHAR2                  IN/OUT
FUNCTION ROLLUP_ON_ORDER RETURNS NUMBER
Argument Name                  Type                      In/Out Default?
------------------------------ ------------------------ ------ --------
O_ERROR_MESSAGE                VARCHAR2                  IN/OUT
------------------------------ ------------------------ ------ --------
ROLLUP_ALLOC_OUT (FUNCTION)<return value>  NUMBER      OUT
ROLLUP_ALLOC_OUT
O_ERROR_MESSAGE                VARCHAR2                  IN/OUT
------------------------------ ------------------------ ------ --------
```

```
ROLLUP_CUSTOMER_ORDER (FUNCTION) <return value>  NUMBER   OUT
ROLLUP_CUSTOMER_ORDER
O_ERROR_MESSAGE                VARCHAR2                IN/OUT
------------------------------ ---------------------- ------ --------
```

There are six functions in the package. Each of the four functions in the package should be called by its own batch program.

- AlcSnapshotSOH
- AlcSnapshotAllocIn
- AlcSnapshotOnOrder
- AlcSnapshotCrosslink
- AlcSnapshotAllocOut
- AlcSnapshotCustomerOrder

Implementation

There are six different batch executables, each executable calling the appropriate method from the above ALC_HIER_LVL_INV_SNAPSHOT_SQL package. Clarifications on the batch functionality:

- Each batch should state success or failure, whether an exception is caught or not.
- There is no need for restart recovery, intermittent commits, or threading.
- Login validation standard logic used by the scheduled alloc program should be applied here as well.
- The programs are run sequentially. The correct order is documented in the Merchandising batch schedule and controlled by whichever scheduling tool used at a particular customer.
- There are no special security requirements for the program. Any user who can log into the Allocation product can have the ability to run the batch processes.