Oracle® Retail Merchandising

Data Conversion Guide
Release 16.0
E65489-01

December 2016

ORACLE

Oracle® Retail Merchandising Data Conversion Guide, Release 16.0

E65489-01

Primary Author: Randy Kapelke

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

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

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

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

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

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

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

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your beta trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Contents

Send Us Your Comments

Oracle Retail Merchandising, Data Conversion Guide, Release 15.0

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 www.oracle.com.

Preface

This *Oracle Retail Merchandising Data Conversion Operations Guide* is a reference for the data conversion operations required to migrate from legacy retail management systems to the Oracle Retail Merchandising software.

This guide describes the data conversion operations that begin with flat files produced from the databases of legacy applications. It details the content and format of each flat file required to perform the data conversion, as well as the tables created and populated by the conversion scripts.

Audience

The *Oracle Retail Merchandising Data Conversion Operations Guide* is intended for the Oracle Retail Merchandising Operations Management applications integrators and implementation staff, as well as the retailer's IT personnel.

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following:

- *Oracle Retail Merchandising System documentation set*
- *Oracle Retail Price Management documentation set*

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

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

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

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

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

Conventions

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

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

1

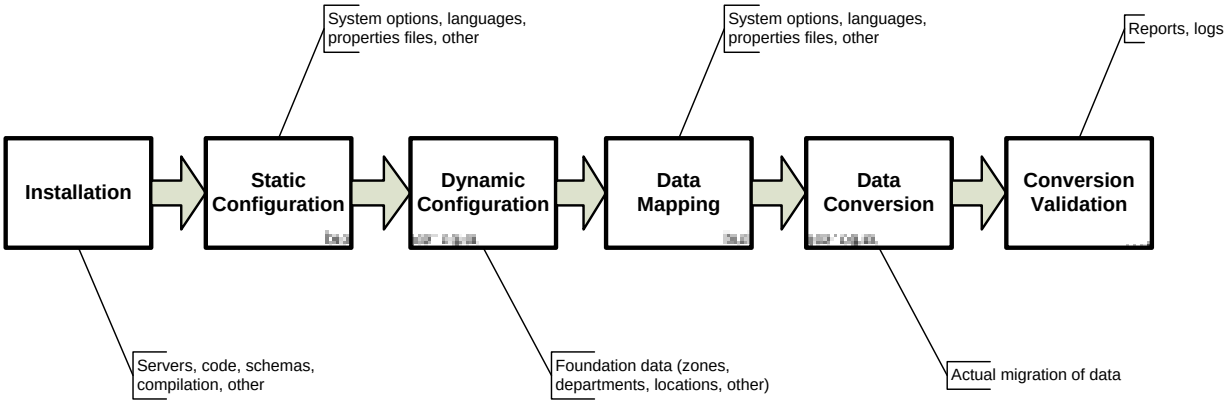
Data Conversion Overview

This chapter gives a brief introduction to the overall process to convert legacy data to the tables required by the Oracle Retail Merchandising applications. You can perform the conversion using a data conversion toolset designed specifically for this purpose.

This chapter describes the components of the data conversion toolset and the sequence of data conversion. It also notes some basic assumptions and prerequisites for performing the data conversion.

Data Conversion Process

The conversion processing performed with the data conversion toolset is one part of the total scope of the migration from legacy systems to the Merchandising applications.



Data Conversion Process

Before actual data conversion begins, the implementation team must complete analysis, mapping, preparation, and extraction of the legacy data into the flat files required for conversion. The Oracle Retail implementation team performs this work with the retailer's systems management, database management, systems analysis, and operations staff.

The data conversion toolset assumes clean data that conforms to the data structures detailed in this guide.

Data Conversion Approach

The overall approach of this data conversion toolset is to use one data loading script for each table or functional area, based on the input files provided by the legacy system. These scripts are executed in sequence and can be scheduled on the batch scheduler as a onetime run during implementation. This toolset assumes that all data loaded is 'clean', so there is no data validation during data load. If there are any issues with data during conversion load, you must manually view log files to determine the cause and correct the data.

The following scripts are included in this data conversion toolset:

- Configuration scripts: These scripts describe configuration and setup tasks required before you can use the data conversion toolset.
- Load scripts: For the identified functional areas, there are specific RMS tables to be loaded with the converted data. This effort can be achieved through individual shell scripts, one each for a table. These shell scripts load data in a two-step process: first populating a staging table from the flat file by using SQLLOADER, to be followed by the next step, from the staging tables to RMS tables by calling a load function.

  Note: The approach for restart and recovery follows.

  1. If there is a failure when loading into a staging table, then the batch fails. Once the batch is re-run, the data in the table from the previous failed run is truncated and the corrected file is reloaded.

  2. If there is a failure when loading into RMS table from the staging table, the batch fails again. Because the commit at the end was not triggered, the records inserted until the failure are rolled back. After this, the batch must be run, and the first step above initiates.

- Control files: Control files contain both the staging table name and the table column definition. This is a direct mapping with the actual flat file. The Load script calls SQLLOADER which references the corresponding control files to appropriately load the flat file data to the staging table.

  Note: Before you begin using the data conversion toolset, you must install the Merchandising applications and load all seed data. For more information, see Appendix: Seed Data Installation.

Prerequisites and Assumptions

The following prerequisites and assumptions apply to the data conversion processes described in this guide:

- Transformation of legacy data is not included as part of the data conversion toolset. Data loaded in flat files must be a clean data. There is no data validation included in this toolset.
- Database constraints must be turned off.
- All database triggers must be evaluated to determine which need to be turned off during the conversion effort.

How to Use This Guide

This guide describes available conversion auto loading programs and processing involved.

There are functional and technical descriptions of all programs included in the data conversion toolset. The program descriptions are organized by functional areas:

- Core
- Merchandise Hierarchy
- Organizational Hierarchy

- Suppliers
- Items
- Multiple Set of Books
- Optional Data

> **Note:** Data conversion must be performed in order by functional area, according to the organization of this guide and the prerequisites stated for each functional area.

The description of each program includes the following information:

- Program purpose and functionality
- Technical specifications
- Field level definitions
- Flat file layouts

To perform data conversion, follow this guide starting with Chapter 2. This guide has the following chapters:

Chapter		Description
2	Configuration Script	This chapter describes configuration and setup tasks required before you can use the data conversion toolset. It also details how to customize the toolset for your specific data conversion needs.
3 4 5 6 7 8	Core Merchandise Hierarchy Organizational Hierarchy Suppliers Items Multiple Sets of Books	These chapters describe in detail all the programs and files required to load data for each of the functional areas. Each chapter also contains a Prerequisites section that lists all tasks that must be completed prior to running the tools for that functional area. Some chapters also have a Post-Loading Requirements section that describes tasks that must be done before data conversion is considered complete for that functional area.
9	Optional Data	This chapter describes additional optional data that you can load manually for each of the functional areas. Optional data can be loaded after auto-loading is complete.
Appendix	Seed Data Installation	This appendix describes the scripts used to load seed data at the time of installation.

2

Configuration Script

Configuration File Definition

The configuration file (DC_LOAD.CFG) contains all directory paths for the executable KSH scripts, SQL scripts, and so on. It also contains log and temp directories needed by the load scripts. The directories are stored in variables accessible to the calling script through the export command. This configuration script must be set up prior to start of data conversion runs.

> **Note:** Before you begin using the data conversion toolset, you must install the merchandising applications and load all seed data. For more information, see Appendix: Seed Data Installation.\

Directories

Create a separate set of UNIX directories to hold the data conversion toolset components. The following directories specific to data conversion should be configured before running the master script. All data and log directories must have read and write privileges.

Directory	Name	Description
Data directory	dataDir	This directory contains the data files (*.dat) used to load information to the staging tables.
Data completed directory	dataCompDir	This directory contains processed data files.
Bad file directory	badDir	This directory contains files with rejected records (*.bad files).
Discard file directory	dscDir	This directory contains discarded files (*.dsc). This directory can be the same as the bad file directory.
Script directory	scriptDir	This directory contains all the scripts and control files used in the conversion process.
Log directory	logDir	This directory contains the conversion script execution logs.
Status directory	statusDir	This directory contains the status files created after each load. This directory can be the same as the log directory.
RMS bin directory	rmsBinDir	This is the directory where the RMS batch executables are installed.

Other directories can be created as needed.

Variables

The following variables are shared across all conversion scripts:

Variable	Description
connectStr	Oracle Wallet alias to be provided to connect to the database.
dataExt	File extension for data files, default .dat.
badExt	File extension for bad files, default .bad.
dscExt	File extension for discard files, default .dsc.
statusExt	File extension for status files, default .status.
seqExt	File extension for sequence files, default .seq.

Other variables are used as needed.

Library File Description

The Library file (dc_load.lib) is a collection of common functions used by the load scripts used in the conversion process. The functions are as follows:

- **checkCfg** - This function is called in by the load scripts to check whether the configuration file contains sufficient information to run the conversion load.
- **checkError** - This function is called inside execKsh and execSql, after executing the script read from the sequence file. It checks the process status and writes the message to the log file.
- **checkFile** - This function checks whether the file passed in meets certain file attribute conditions. Using options, this function checks the following:
  - File existence (option –e)
  - Read access (option –r)
  - File size (if greater than 0, using option -s)
  - If executable (option –x)

  For conversion files defined in the configuration file, attribute checks are performed according to type:
  - For data (*.dat) files, files are checked for existence, size, and read access.
  - For bad (*.bad), discard (*.dsc), and status (*.status) files; only existence is checked.

  This function is also used to check whether a program is executable. It returns 1 if one of the set conditions fails.
- **getAvailThread** - This function gets the minimum thread value for the passed-in batch program from the restart_program_status table where the status is 'ready for start'. It uses an embedded SQL SELECT to get the information.
- **refreshThreads** - This function updates the restart_program_status table to 'ready for start' status for threads the previously completed successfully.
- **execPgm** - This function is called from the main script to execute KSH or other executable scripts. The program file passed in is verified first, prior to execution, using the checkFile function. If the file is valid, the script is invoked as it would be in the command line. All messages displayed by the called script are written to the log.
- **execSql** - This function is called from the main script to execute SQL scripts only, as read from the *.seq file. The SQL file passed in is verified first, prior to execution,

using the checkFile function. The sqlplus –s command is used to execute the SQL script, passing the connect string defined in the env file and the script name. All messages displayed by the called script are written to the log file.

- **execRms** - This function is called from the main script to execute RMS batch programs, threaded according to the restart control tables. Because this script allows execution of custom RMS batch programs, the function checks whether the file is valid, using the checkFile function with the option –x. If no path is defined, it uses the default directory for the RMS executables defined in the load configuration file. If the file is found to be valid, the function checks the type of batch program it will run.

  For prepost batch, the function extracts the main batch and the prepost indicator from the seqData2 information and executes the batch.

  For file- or table-based batch programs, the function uses more complex logic to take advantage of the multi-threading capabilities of the batch. File-based programs are dependent on input files to load information to the RMS tables. The script checks whether at least an input file exists. If so, the script loops through the file list, refreshes the restart_program_status table using the refreshThreads function, and attempts to get an available thread using the getAvailThread function. If a thread is found, it moves the input file to a process directory (defined in the *.cfg file), appends the thread number, and executes the batch. These steps are repeated until all files in the input file directory (also defined in the *.cfg file) are processed. Only files with the correct file prefix (for example, POSU for posupld files) are processed.

  For table-based batch programs, the function checks whether a driver value is defined. If none is defined, the batch is not threaded, or it is threaded using its parameters. In this case, the function checks the seqData2 information passed in to the function. If seqData2 contains no data, the batch is executed immediately. If the parameter variable (from the seqData2 value) contains information, the function builds a parameter list (paramLst array) and loops through the parameter list. If the parameter list has values, the script starts the processing by obtaining an available thread through the refreshThreads and getAvailThread functions, and executing the batch by passing the parameter values required. Table-based batch programs are handled by obtaining the number of threads from the restart control, refreshing the threads, and looping through each available thread.

  Simultaneous execution of the batch (multithreading) is achieved through a sub process (& appended to the batch execution line).

- **execBRPgm** - The scripts are only executed if the system's base country is BR and it's localized. The program file passed in is verified first, prior to execution, using the checkFile function. If the file is valid, the script is invoked as it would be in the command line. All messages displayed by the called script are written to the log.

3

Core

This chapter describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- TERMS_HEAD
- TERMS_DETAIL
- FREIGHT_TYPE
- FREIGHT_TERMS
- FREIGHT_SIZE
- VAT_CODES
- VAT_CODE_RATES
- VAT_REGION
- UDA
- UDA_VALUES
- TICKET_TYPE_HEAD
- TICKET_TYPE_DETAIL
- DIFF_IDS
- TSF_ENTITY
- FIF_GL_SETUP
- ORG_UNIT
- BRAND

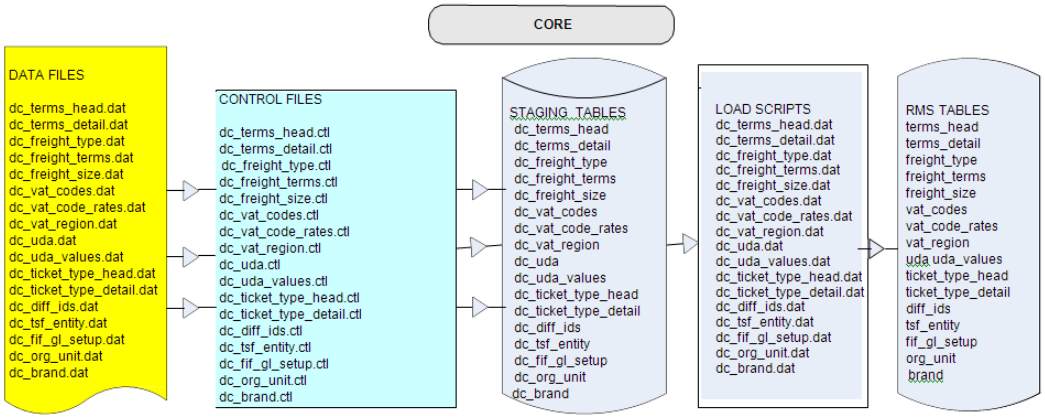
The following programs are included in this functional area:

- Load scripts
  - dc_terms_head.ksh
  - dc_terms_detail.ksh
  - dc_freight_type.ksh
  - dc_freight_terms.ksh
  - dc_freight_size.ksh
  - dc_vat_codes.ksh
  - dc_vat_code_rates.ksh
  - dc_vat_region.ksh
  - dc_uda.ksh
  - dc_uda_values.ksh
  - dc_ticket_type_head.ksh
  - dc_ticket_type_detail.ksh
  - dc_diff_ids.ksh
  - dc_tsf_entity.ksh
  - dc_fif_gl_setup.ksh
  - dc_org_unit.ksh
  - dc_brand.ksh

- Control files
  - dc_terms_head.ctl
  - dc_terms_detail.ctl
  - dc_freight_type.ctl
  - dc_freight_terms.ctl
  - dc_freight_size.ctl
  - dc_vat_codes.ctl
  - dc_vat_code_rates.ctl
  - dc_vat_region.ctl
  - dc_uda.ctl
  - dc_uda_values.ctl
  - dc_ticket_type_head.ctl
  - dc_ticket_type_detail.ctl
  - dc_diff_ids.ctl
  - dc_tsf_entity.ctl
  - dc_fif_gl_setup.ctl
  - dc_org_unit.ctl
  - dc_brand.ctl

Data Flow

The following diagram shows the data flow for the Core functional area:



Data Flow for Core Functional Area

Prerequisites

Before you begin using the data conversion toolset for the Core functional area, there are tables that must be loaded manually, because of data dependencies for auto-loading within this functional area. Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

> **Note:** It is assumed that you have already installed the Merchandising applications and loaded all installation seed data. For more information, see Appendix: Seed Data Installation.

Table	Required / Optional / Comments
CALENDAR	Required Note: Calendar data is loaded as part of installation; however, the data provided may not match the calendar that fits your business operation. Consider revising the calendar data script. Tip: CALENDAR.MONTH_454 = 1 is January (not fiscal year).
HALF	Required
BANNER	Required when multi-channel is turned on
CHANNELS	Required
SEASONS	Optional
PHASES	Optional
DIFF_TYPE	Required
TSFZONE	Required
STORE_FORMAT	Required
BUYER	Optional
MERCHANT	Optional
CVB_HEAD	Optional
CVB_DETAIL	Optional
ELC_COMP	Required only if up charges are loaded
STATE	Required only if using addresses in U.S. locations

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the Staging Table Definition columns Field Name and Data Type (including length) define the physical staging table.

Terms

DC_TERMS_HEAD Table

File name: **DC_TERMS_HEAD.DAT**

Control File: **DC_TERMS_HEAD.CTL**

Staging table: **DC_TERMS_HEAD**

Suggested post-loading validation:

- Ensure that TERMS_HEAD.TERMS is unique.
- Capture the count from TERMS_HEAD and compare to flat file DC_TERMS_HEAD.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TERMS	Alpha-numeric	15	Y	Unique identifier of supplier payment terms record.	TERMS	VARCHAR2(15)
TERMS_CODE	Alpha-numeric	50	Y	Code that represents the supplier payment terms in Oracle Financials.	TERMS_CODE	VARCHAR2(50)
TERMS_DESC	Alpha-numeric	240	Y	Description of the supplier payment terms. For example, 2.5% 30 Days.	TERMS_DESC	VARCHAR2(240)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
RANK	Alpha-numeric	10	Y	Rank to rate invoice payment terms against purchase order terms. These rankings are defined in the retailer's financial system. These rankings are used in "best terms" calculation. When terms are compared, the term with the higher rank (meaning lower number - 1 is the highest rank) is the best term. This must be a whole number greater than zero.	RANK	NUMBER(10)

DC_TERMS_DETAIL Table

File name: **DC_TERMS_DETAIL.DAT**

Control File: **DC_TERMS_DETAIL.CTL**

Staging table: **DC_TERMS_DETAIL**

Suggested post-loading validation:

- Ensure that TERMS_DETAIL.TERMS is a valid TERMS_HEAD.TERMS.
- Ensure that each combination of TERMS_DETAIL.TERMS and TERMS_DETAIL.TERMS_SEQ is unique.
- Capture the count from TERMS_DETAIL and compare to flat file DC_TERMS_DETAIL.DAT to ensure that all rows are loaded.

> **Note:** Column order for this file does not match the RMS TERMS_DETAIL table.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TERMS	Alpha-numeric	15	Y	Unique identifier of supplier payment terms record. This ties the detail record to the appropriate dc_terms_head record.	TERMS	VARCHAR2(15)

FILE FORMAT					STAGING TABLE DEFINITION	
TERMS_SEQ	Numeric	10	Y	Order sequence in which to apply the discount percent.	TERMS_SEQ	NUMBER(10)
DUEDAYS	Numeric	3	Y	Number of days until payment is due.	DUEDAYS	NUMBER(3,)
DUE_MAX_AMOUNT	Numeric	12,4	Y	Maximum payment amount due by a given date.	DUE_MAX_AMOUNT	NUMBER(12,4)
DUE_DOM	Numeric	2	Y	Day of month used to calculate due date of invoice payment line. For example, 1 represents the 1st day of the month.	DUE_DOM	NUMBER(2)
DUE_MM_FWD	Numeric	3	Y	Number of months ahead used to calculate due date of invoice payment line.	DUE_MM_FWD	NUMBER(3)
DISCDAYS	Numeric	3	Y	Number of days in which payment must be made in order to receive the discount.	DISCDAYS	NUMBER(3)
PERCENT	Numeric	12,4	Y	Percent of discount if payment is made within specified time period.	PERCENT	NUMBER(12,4)
DISC_DOM	Numeric	2	Y	Day of month used to calculate discount date of invoice payment line. For example, 1 represents the 1st day of the month.	DISC_DOM	NUMBER(2)
DISC_MM_FWD	Numeric	3	Y	Number of months ahead used to calculate discount date of invoice payment line.	DISC_MM_FWD	NUMBER(3)
ENABLED_FLAG	Alpha-numeric	1	Y	Indicates whether payment terms are valid or invalid within the application.	ENABLED_FLAG	VARCHAR2(1)
CUTOFF_DAY	Numeric	2	Y	Day of the month after which Oracle Payables schedules payment using the day after the current month.	CUTOFF_DAY	NUMBER(2)
FIXED_DATE	Alpha-numeric	7	N	Fixed due date.	FIXED_DATE	DATE

FILE FORMAT					STAGING TABLE DEFINITION	
START_DATE_ACTIVE	Alpha-numeric	7	N	Date in which the payment terms become active.	START_DATE_ACTIVE	DATE
END_DATE_ACTIVE	Alpha-numeric	7	N	Date in which the payment terms become inactive.	END_DATE_ACTIVE	DATE

Freight

DC_FREIGHT_TYPE Table

File name: **DC_FREIGHT_TYPE.DAT**

Control File: **DC_FREIGHT_TYPE.CTL**

Staging table: **DC_FREIGHT_TYPE**

Suggested post-loading validation

- Ensure that FREIGHT_TYPE.FREIGHT_TYPE is unique.
- Capture the count from FREIGHT_TYPE and compare to flat file DC_FREIGHT_TYPE.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req Ind**	**Description**	**Field Name**	**Data Type**
Freight_Method	Alpha-numeric	6	Y	Unique identifier of the freight method.	FREIGHT_TYPE	VARCHAR2(6)
Freight_Method_Desc	Alpha-numeric	250	Y	Description of the freight method. Examples are Full Container Load and Less than Container Load.	FREIGHT_TYPE_DESC	VARCHAR2(250)

DC_FREIGHT_TERMS Table

File name: **DC_FREIGHT_TERMS.DAT**

Control file: **DC_FREIGHT_TERMS**

Staging table: **DC_FREIGHT_TERMS**

Suggested post-loading validation

- Ensure that FREIGHT_TERMS.FREIGHT_TERMS is unique.
- Capture the count from FREIGHT_TERMS and compare to flat file DC_FREIGHT_TERMS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
FREIGHT_TERMS	Alpha-numeric	30	Y	Unique identifier of freight terms record.	FREIGHT_TERMS	VARCHAR2(30)
TERM_DESC	Alpha-numeric	240	Y	Description of the freight terms. Examples include a percentage of total cost or a flat fee.	TERM_DESC	VARCHAR2(240)
START_DATE_ACTIVE	Alpha-numeric	9	N	Date on which the freight terms become active. Date format is DDMONYYYY (for example, 02JAN2011).	START_DATE_ACTIVE	DATE
END_DATE_ACTIVE	Alpha-numeric	9	N	Date on which the freight terms become inactive. Date format is DDMONYYYY.	END_DATE_ACTIVE	DATE
ACTIVE_FLAG	Alpha-numeric	1	Y	Indicates whether freight terms are valid or invalid within the application. Default = N.	ENABLED_FLAG	VARCHAR2(1)

DC_FREIGHT_SIZE Table

File name: **DC_FREIGHT_SIZE.DAT**

Control File: **DC_FREIGHT_TERMS.CTL**

Staging table: **DC_FREIGHT_SIZE**

Suggested post-loading validation

- Ensure that FREIGHT_SIZE.FREIGHT_SIZE is unique.
- Capture count from FREIGHT_SIZE and compare to flat file DC_FREIGHT_SIZE.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
FREIGHT_SIZE	Alpha-numeric	6	Y	Unique identifier of the freight size record.	FREIGHT_SIZE	VARCHAR2(6)
FREIGHT_SIZE_DESC	Alpha-numeric	250	Y	Freight size description (for example, 40 foot container).	FREIGHT_SIZE_DESC	VARCHAR2(250)

VAT

DC_VAT_CODES Table

File name: **DC_VAT_CODES.DAT**

Control File: **DC_VAT_CODES.CTL**

Staging table: **DC_VAT_CODES**

Suggested post-loading validation:

- Ensure that VAT_CODES.VAT_CODE is unique.
- Capture the count from VAT_CODES and compare to flat file DC_VAT_CODES.DAT to ensure that all rows are loaded.
- VAT-related tables are only inserted if vat_ind on system_options is Y and default_tax_type is not GTAX (SVAT is used).

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
VAT_CODE	Alpha-numeric	6	Y	Unique identifier of value added tax code, used to determine which items are subject to VAT. For example, the Valid values includes: S – Standard Z – Zero	VAT_CODE	VARCHAR2(6)
VAT_CODE_DESC	Alpha-numeric	120	Y	Value added tax code description.	VAT_CODE_DESC	VARCHAR2(120)

DC_VAT_CODE_RATES Table

File name: **DC_VAT_CODE_RATES.DAT**

Control file: **DC_VAT_CODE_RATES.CTL**

Staging table: **DC_VAT_CODE_RATES**

Suggested post-loading validation:

- Ensure that VAT_CODE_RATES.VAT_CODE is a valid VAT_CODES.VAT_CODE.
- Capture the count from VAT_CODE_RATES and compare to flat file DC_VAT_CODE_RATES.DAT to ensure that all rows are loaded.
- VAT-related tables are only inserted if vat_ind on system_options is Y and default_tax_type is not GTAX (SVAT is used).

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
VAT_CODE	Alpha-numeric	6	Y	Unique identifier of value added tax code. This ties the record to the appropriate dc_vat_codes record.	VAT_CODE	VARCHAR2(6)
ACTIVE_DATE	Alpha-numeric	9	Y	Date on which VAT rate becomes active. Date format is DDMONYYYY (for example, 02JAN2011).	ACTIVE_DATE	DATE
VAT_RATE	Numeric	20,10	Y	VAT rate as a percentage.	VAT_RATE	NUMBER(20,10)

DC_VAT_REGION Table

File name: **DC_VAT_REGION.DAT**

Control file: **DC_VAT_REGION.CTL**

Staging table: **DC_VAT_REGION**

Suggested post-loading validation:

- Ensure that VAT_REGION.VAT_REGION is unique.
- Capture the count from VAT_REGION and compare to flat file DC_VAT_REGION.DAT to ensure that all rows are loaded.
- VAT-related tables are only inserted if vat_ind on system_options is Y and default_tax_type is not GTAX (SVAT is used).

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
VAT_REGION	Numeric	4	Y	Unique identifier of VAT region. VAT region is determined by the VAT authority.	VAT_REGION	NUMBER(4)
VAT_REGION_NAME	Alpha-numeric	120	Y	Name/description of the VAT region.	VAT_REGION_NAME	VARCHAR2(120)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
VAT_REGION_TYPE	Alpha-numeric	6	Y	VAT region type. Valid values include E for base EU members, M for EU members and N for non members of the EU.	VAT_REGION_TYPE	VARCHAR2(6)
ACQUISITION_VAT_IND	Alpha-numeric	1	Y	Indicates if acquisition VAT is applicable to the VAT region. Valid values are Y and N.	ACQUISITION_VAT_IND	VARCHAR2(1)
REVERSE_VAT_THRESHOLD	Numeric	20, 4	N	This holds the invoice-level total value limit. The reverse charge VAT rule only applies if the total value of items are subject to reverse charge VAT exceeds the threshold for an invoice. This value is expressed in the country currency of the vat_region, which typically only belongs to one country.	REVERSE_VAT_THRESHOLD	NUMBER(20, 4)

UDA

DC_UDA Table

File name: **DC_UDA.DAT**

Control file: **DC_UDA.CTL**

Suggested post-loading validation:

- Ensure that UDA.UDA_ID is unique.
- Capture the count from UDA and compare to flat file DC_UDA.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
UDA_ID	Numeric	5	Y	Unique identifier of user-defined attribute.	UDA_ID	NUMBER(5)
UDA_DESC	Alpha-numeric	120	Y	Description of user-defined attribute. UDAs do not have specific processing within RMS.	UDA_DESC	VARCHAR2(120)
DISPLAY_TYPE	Alpha-numeric	2	Y	How the UDA displays to the user online. Valid values are: LV - List of values. FF - Free-form text. DT – Date. Note: A UDA with DISPLAY_TYPE LV must also have a corresponding UDA record in DC_UDA_VALUES.DAT.	DISPLAY_TYPE	VARCHAR2(2)
DATA_TYPE	Alpha-numeric	12	N	Data type associated with the UDA. If display_type =DT, the data_type should be DATE. If no value is provided in the flat file, the default value is DATE. If display_type = FF, the data_type should be ALPHA. If no value is provided in the flat file, the default value is ALPHA. If display_type = LV, the data_type can either be NUM or ALPHA. If no value is provided in the flat file, the default value is ALPHA.	DATA_TYPE	VARCHAR2(12)
DATA_LENGTH	Numeric	3	N	Maximum length of the UDA values. This field should not be populated for a DT display type. It is optional for FF and LV display types. For LV, this constrains what is stored in UDA_VALUES. UDA_VALUE_DESCRIPTION. For FF, this constrains what is stored in UDA_ITEM_FF, UDA_TEXT.	DATA_LENGTH	VARCHAR2(3)
SINGLE_VALUE_IND	Alpha-numeric	1	Y	Indicates whether this UDA can have only one value associated with an item. If Y, only one value of the UDA can be associated with an item.	SINGLE_VALUE_IND	VARCHAR2(1)

DC_UDA_VALUES Table

File name: **DC_UDA_VALUES.DAT**

Control file: **DC_UDA_VALUES.CTL**

Staging table: **DC_UDA_VALUES**

Suggested post-loading validation

- Ensure that UDA_VALUES.UDA_ID is a valid UDA.UDA_ID.
- Capture the count from UDA_VALUES and compare to flat file DC_UDA_VALUES.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
UDA_ID	Numeric	5	Y	Unique identifier of user-defined attribute. This applies only to UDAs with LV display type. This ties the record to the appropriate dc_uda record.	UDA_ID	NUMBER(5)
UDA_VALUE_DESC	Alpha-numeric	250	Y	Description of the UDA value.	UDA_VALUE_DESC	VARCHAR2(250)

Ticket Type

DC_TICKET_TYPE_HEAD Table

File name: **DC_TICKET_TYPE_HEAD.DAT**

Control file: **DC_TICKET_TYPE_HEAD.CTL**

Staging table: **DC_TICKET_TYPE_HEAD**

Suggested post-loading validation:

- Ensure that TICKET_TYPE_HEAD.TICKET_TYPE_ID is unique.
- Capture the count from TICKET_TYPE_HEAD and compare to flat file DC_TICKET_TYPE_HEAD.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TICKET_TYPE_ID	Alpha-numeric	4	Y	Unique identifier of ticket or label type.	TICKET_TYPE_ID	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
TICKET_TYPE_DESC	Alpha-numeric	120	Y	Description of ticket or label type.	TICKET_TYPE_DESC	VARCHAR2(120)
SHELF_EDGE_IND	Alpha-numeric	1	Y	Indicates whether this is a shelf edge label. Default = N.	SEL_IND	VARCHAR2(1)

DC_TICKET_TYPE_DETAIL Table

File name: **DC_TICKET_TYPE_DETAIL.DAT**

Control file: **DC_TICKET_TYPE_DETAIL.CTL**

Staging table: **DC_TICKET_TYPE_DETAIL**

Suggested post-loading validation:

- Ensure that TICKET_TYPE_DETAIL.TICKET_TYPE_ID is a valid TICKET_TYPE_HEAD.TICKET_TYPE_ID.
- Ensure that TICKET_TYPE_DETAIL.TICKET_ITEM_ID (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = TCKT.
- Ensure that TICKET_TYPE_DETAIL.UDA_ID (if not NULL) is a valid UDA.UDA_ID.
- Capture the count from TICKET_TYPE_DETAIL and compare to flat file DC_TICKET_TYPE_DETAIL.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TICKET_TYPE_ID	Alpha-numeric	4	Y	Unique identifier of ticket or label type. This ties the record to the appropriate dc_ticket_type_ head record.	TICKET_TYPE_ID	VARCHAR2(4)
TICKET_ITEM_ID	Alpha-numeric	4	N	Identifier of type of information/attribute to be displayed on ticket or label type. Valid values come from the TCKT code_type. If this field is populated, the uda_id field should not be populated.	TICKET_ITEM_ID	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
UDA_ID	Numeric	5	N	If the information to be displayed on the ticket or label type is a user defined attribute (UDA), this field should be populated with the uda_id. This value comes from the uda.uda_id field. If this field is populated, the ticket_item_id field should not be populated.	UDA_ID	NUMBER(5)

Note: If any records are in the BAD or DISCARD file, the RMS table must be truncated and the entire file must be rerun. No new records within a sequence group can be added to the RMS table through the scripts.

Diff IDs – DC_DIFF_IDS Table

File name: **DC_DIFF_IDS.DAT**

Control file: **DC_DIFF_IDS.CTL**

Staging table: **DC_DIFF_IDS**

Suggested post-loading validation:

- Ensure that DIFF_IDS.DIFF_ID is unique.
- Ensure that DIFF_IDS.DIFF_TYPE is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = DIFF.
- Capture the count from DIFF_IDS and compare to flat file DC_DIFF_IDS.DAT to ensure all that rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
DIFF_ID	Alpha-numeric	10	Y	Unique identifier of the differentiator.	DIFF_ID	VARCHAR2(10)
DIFF_TYPE	Alpha-numeric	6	Y	Differentiator group associated to the differentiator. Valid values are from the DIFF_TYPE column in the DIFF_TYPE table. Examples are C for Color and F for Flavor.	DIFF_TYPE	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
DIFF_DESC	Alpha-numeric	120	Y	Description of the differentiator. Examples are Red, Stripe, and Strawberry.	DIFF_DESC	VARCHAR2(120)
INDUSTRY_CODE	Alpha-numeric	10	N	Unique number that represents all possible combinations of sizes, according to the National Retail Federation. Optional.	INDUSTRY_CODE	VARCHAR2(10)
INDUSTRY_SUBGROUP	Alpha-numeric	10	N	Unique number that represents all different color range groups, according to the National Retail Federation. Optional.	INDUSTRY_SUBGROUP	VARCHAR2(10)

TSF Entities - DC_TSF_ENTITY Table

File name: **DC_TSF_ENTITY.DAT**

Control file: **DC_TSF_ENTITY.CTL**

Staging table: **DC_TSF_ENTITY**

Suggested post-loading validation

- Ensure that TSF_ENTITY.TSF_ENTITY_ID is unique.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TSF_ENTITY_ID	Numeric	10	Y	Unique identifier of the transfer entity.	TSF_ENTITY_ID	NUMBER(10)
TSF_ENTITY_DESC	Alpha-numeric	120	Y	Description of the transfer entity.	TSF_ENTITY_DESC	VARCHAR2(120)

Set of Books – DC_TSF_FIF_GL_SETUP Table

File name: **DC_TSF_FIF_GL_SETUP.DAT**

Control file: **DC_FIF_GL_SETUP.CTL**

Staging table: **DC_FIF_GL_SETUP**

Suggested post-loading validation

- Ensure that FIF_GL_SETUP.SET_OF_BOOKS_ID is unique.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
SET_OF_BOOKS_ID	Numeric	15	Y	Unique identifier for the set of books.	SET_OF_BOOKS_ID	NUMBER(10)
LAST_UPDATE_ID	Numeric	15	Y	Last updated ID for transactions.	SET_OF_BOOKS_ID	VARCHAR2(120)
SEQUENCE1_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE1_DESC	VARCHAR2(20)
SEQUENCE2_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE2_DESC	VARCHAR2(20)
SEQUENCE3_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE3_DESC	VARCHAR2(20)
SEQUENCE4_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE4_DESC	VARCHAR2(20)
SEQUENCE5_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE5_DESC	VARCHAR2(20)
SEQUENCE6_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE6_DESC	VARCHAR2(20)
SEQUENCE7_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE7_DESC	VARCHAR2(20)
SEQUENCE8_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE8_DESC	VARCHAR2(20)
SEQUENCE9_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE9_DESC	VARCHAR2(20)
SEQUENCE10_DESC	Alpha-numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE10_DESC	VARCHAR2(20)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
CATEGORY_ID	Numeric	38	Y	Oracle category ID. Default for purchase order feed.	CATEGORY_ID	NUMBER(38)
DELIVER_TO_LOCATION_ID	Numeric	15	Y	Oracle location ID. Default for purchase order feed.	DELIVER_TO_LOCATION_ID	NUMBER(15)
DESTINATION_ORGANIZATION_ID	Numeric	38	Y	Oracle Organization ID Default for purchase order feed.	DESTINATION_ORGANIZATION_ID	NUMBER(38)
PERIOD_NAME	Alpha-numeric	15	N	User entered accounting period name as defined in financial applications.	PERIOD_NAME	VARCHAR2(15)
SET_OF_BOOKS_DESC	Alpha-numeric	120	Y	Set of books description.	SET_OF_BOOKS_DESC	VARCHAR2(120)
CURRENCY_CODE	Alpha-numeric	3	Y	Currency code for the Set of Books ID.	CURRENCY_CODE	VARCHAR2(3)

Organization Unit – DC_ORG_UNIT Table

File name: **DC_ORG_UNIT.DAT**

Control file: **DC_ORG_UNIT.CTL**

Staging table: **DC_ORG_UNIT**

Suggested post-loading validation

- Ensure that **ORG.UNIT.ORG_UNIT_ID** is unique.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ORG_UNIT_ID	Numeric	10	Y	Unique identifier for the Organization ID.	ORG_UNIT_ID	NUMBER(15)
DESCRIPTION	Alpha-numeric	120	Y	Description of the organization unit.	DESCRIPTION	VARCHAR2(120)
SET_OF_BOOKS_ID	Numeric	15	N	Set of books ID.	SET_OF_BOOKS_ID	NUMBER(15)

Brand – DC_BRAND Table

File name: **DC_BRAND.DAT**

Control file: **DC_BRAND.CTL**

Staging table: **DC_BRAND**

Suggested post-loading validation

- Capture the count from BRAND and compare to flat file **DC_BRAND.DAT** to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
BRAND_NAME	Alpha-numeric	30	Y	Brand Name.	BRAND_NAME	VARCHAR2(30)
BRAND_DESCRIPTION	Alpha-numeric	120	Y	Brand Description.	BRAND_DESCRIPTION	VARCHAR2(120)

Load Scripts

DC_TERMS_HEAD.KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TERMS_HEAD staging table.

LOAD_TERMS_HEAD– This function contains a PL/SQL block that selects from the DC_ TERMS_HEAD staging table and inserts the data to the RMS TERMS_HEAD and TERMS_HEAD_TL table. All the columns from the staging table defined above will directly map to the RMS table The fields that are not required are null.

Required File to Load dc_terms_head.dat

ERROR HANDLING

All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT

Follow each insert statement with a commit command.

DC_TERMS_DETAIL.KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS table.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TERMS_DETAIL staging table.

LOAD_TERMS_DETAIL– This function contains a PL/SQL block that selects from the DC_ TERMS_DETAIL staging table and inserts the data to the RMS TERMS_DETAIL

table. All the columns from the staging table defined above will directly map to the RMS table. The fields that are not required are null.

Required file to load: dc_terms_detail.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_FREIGHT_TYPE .KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_FREIGHT_TYPE staging table.

LOAD_ FREIGHT_TYPE – This function contains a PL/SQL block that selects from the DC_FREIGHT_TYPE staging table and inserts the data to the RMS FREIGHT_TYPE and FREIGHT_TYPE_TL table. All the columns from the staging table defined above will directly map to the RMS table and all are required.

Required file to load: dc_freight_type.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_FREIGHT_TERMS.KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_FREIGHT_TERMS staging table.

LOAD_FREIGHT_TERMS– This function contains a PL/SQL block that selects from the DC_FREIGHT_TERMS staging table and inserts the data to the RMS FREIGHT_TERMS and FREIGHT_TERMS_TL table. All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined).The function returns a Boolean value.

Required file to load: dc_freight_terms.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_FREIGHT_SIZE .KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_FREIGHT_SIZE staging table.

LOAD_FREIGHT_SIZE– This function contains a PL/SQL block that selects from the DC_FREIGHT_SIZE staging table and inserts the data to the RMS FREIGHT_SIZE and FREIGHT_SIZE_TL table. All the columns from the staging table defined above will directly map to the RMS table and are required.

Required file to load: dc_freight_size.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_VAT_CODES.KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_VAT_CODES staging table.

LOAD_VAT_CODES– This function contains a PL/SQL block that selects from the DC_VAT_CODES staging table and inserts the data to the RMS VAT_CODES table if vat_ind on system_options is 'Y' and default_tax_type is NOT 'GTAX' (i.e. SVAT is used.). All the columns from the staging table defined above will directly map to the RMS table and are required.

Required file to load: dc_vat_codes.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_VAT_CODE_RATES.KSH

This ksh script will be called to serve two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_VAT_CODE_RATES staging table.

LOAD_VAT_CODE_RATES– This function contains a PL/SQL block that selects from the DC_VAT_CODE_RATES staging table and inserts the data to the RMS VAT_CODE_RATES table if vat_ind on system_options is 'Y' and default_tax_type is NOT 'GTAX' (i.e. SVAT is used.). All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS tables if no information is provided in the data file (staging table field value is NULL or not defined).

DC_VAT_CODE_RATES to VAT_CODE_RATES Column Defaults

Field Names (RMS Table)	Default Value	Comments
CREATE_DATE	SYSDATE	Date the record was created in RMS. This defaults to the system date
CREATE_ID	Oracle User	User who created the record in RMS. This defaults to the Oracle User

Required file to load: dc_vat_codes.dat (required to verify if vat_codes was loaded previously), **dc_vat_code_rates.dat**

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_VAT_REGION.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_VAT_REGION staging table.

LOAD_VAT_REGION – This function contains a PL/SQL block that selects from the DC_VAT_REGION staging table and inserts the data to the RMS VAT_REGION table if vat_ind on system_options is 'Y' and default_tax_type is NOT 'GTAX' (i.e. SVAT is used.). All the columns from the staging table defined above will directly map to the RMS table and all are required.

Required file to load: dc_vat_region.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UDA.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA staging table.

LOAD_UDA– This function contains a PL/SQL block that selects from the DC_UDA staging table and inserts the data into the RMS UDA table. All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined).

DC_UDA to UDA Column Defaults

Field Name (RMS Table)	Default Value	Comments
MODULE	ITEM	Functional area of RMS to which this applies. The only valid value is ITEM.
DATA_TYPE	ALPHA (FF/LV*) DATE (DT) NUM(LV*)	If display_type =DT, the data type should be DATE. If no value is provided in the flat file, the default value is DATE. If display_type = FF, the data_type should be ALPHA. If no value is provided in the flat file, the default value is ALPHA. If display_type = LV, the data_type can either be NUM or ALPHA. If no value is provided in the flat file, the default value is ALPHA.

Required file to load: dc_uda.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UDA_VALUES.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables. and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA_VALUES staging table.

LOAD_UDA_VALUES– This function contains a PL/SQL block that selects from the DC_UDA_VALUES staging table and inserts the data into the RMS UDA_VALUES table. UDA_VALUES should contain information if the data_type value in the UDA table is LV. All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined).

DC_UDA_VALUES to UDA_VALUES Column Defaults

Field Name (RMS Table)	Default Value	Comments
UDA_VALUE	Sequence generated	Per UDA_ID

Required file to load: dc_uda.dat (required to check if UDA is loaded previously)**,dc_uda_values.dat**

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_TICKET_TYPE_HEAD.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TICKET_TYPE_HEAD staging table.

LOAD_TICKET_TYPE_HEAD– This function contains a PL/SQL block that selects from the DC_TICKET_TYPE_HEAD staging table and inserts the data into the RMS TICKET_TYPE_HEAD table. All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined).

Required file to load: dc_ticket_type_head.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_TICKET_TYPE_DETAIL.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TICKET_TYPE_DETAIL staging table.

LOAD_TICKET_TYPE_DETAIL– This function contains a PL/SQL block that selects from the DC_TICKET_TYPE_DETAIL staging table and inserts the data into the RMS TICKET_TYPE_DETAIL table. All the columns from the staging table defined above will directly map to the RMS table. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined). There should be a value in ticket_item_id or uda_id and not both.

DC_TICKET_TYPE_DETAIL to TICKET_TYPE_DETAIL Column Defaults

Field Name (RMS Table)	Default Value	Comments
SEQ_NO	Sequence generated	Per TICKET_TYPE_ID

Required file to load: dc_ticket_type_head.dat (required to check if TICKET_TYPE_HEAD is loaded previously), **dc_ticket_type_detail.dat**

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_DIFF_IDS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_DIFF_IDS staging table.

LOAD_DIFF_IDS– This function contains a PL/SQL block that selects from the DC_DIFF_IDS staging table and inserts the data to the RMS DIFF_IDS table. All the columns from the staging table defined above will directly map to the RMS table and all are required. The table below defines the default value in the RMS table if no information is provided in the data file (staging table field value is NULL or not defined).

DC_DIFF_IDS to DIFF_IDS Column Defaults

Field Name (RMS Table)	Default Value	Comments
CREATE_DATETIME	sysdate	Date/time the record was created in RMS. This defaults to the system date.
LAST_UPDATE_ID	Oracle User	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	sysdate	Date/time the record was last modified in RMS. This defaults to the system date.

Required file to load: dc_diff_ids.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_TSF_ENTITY.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TSF_ENTITY staging table.

LOAD_TSF_ENTITY– This function contains a PL/SQL block that selects from the DC_TSF_ENTITY staging table and inserts the data to the RMS TSF_ENTITY table. All the columns from the staging table defined above will directly map to the RMS table and all are required.

Required file to load: dc_tsf_entity.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_FIF_GL_SETUP.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_FIF_GL_SETUP staging table.

LOAD_TERMS_HEAD– This function contains a PL/SQL block that selects from the DC_FIF_GL_SETUP staging table and inserts the data to the RMS FIF_GL_SETUP table. All the columns from the staging table defined above will directly map to the RMS table,

SET_OF_BOOKS_ID.LAST_UPDATE_ID, DELIVER_TO_LOCATION_ID, DESTINATION_ORGANIZATION_ID, SET_OF_BOOKS_ID AND CURRENCY_CODE all are required.

Required file to load: dc_fif_gl_setup.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ORG_UNIT.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ORG_UNIT staging table.

LOAD_ORG_UNIT– This function contains a PL/SQL block that selects from the DC_ORG_UNIT staging table and inserts the data to the RMS ORG_UNIT table. All the columns from the staging table defined previously map directly to the RMS table. All fields are required.

Required file to load: dc_org_unit.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_BRAND.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_BRAND staging table.

LOAD_BRAND– This function contains a PL/SQL block that selects from the DC_BRAND staging table and inserts the data to the BRAND table. All the columns from the staging table defined previously map directly to the RMS table. All fields are required.

Required file to load: dc_brand.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Post-Loading Requirements

After using the data conversion toolset for this functional area, there are additional tables that must be loaded manually before you proceed with data conversion for subsequent functional areas, because of data dependencies.

Manual data loading can be performed online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following table lists the tables that require manual data loading and indicates whether each table is required or optional:

Table	Required / Optional
DIFF_GROUP_HEAD	Required
DIFF_GROUP_DETAIL	Required
DIFF_RANGE_HEAD	Optional
DIFF_RANGE_DETAIL	Optional

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts are executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_terms_head.ksh
```

> **Note** the use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

4

Merchandise Hierarchy

This chapter describes data conversion for the following RMS/RPM tables, listed in the order that they must be loaded:

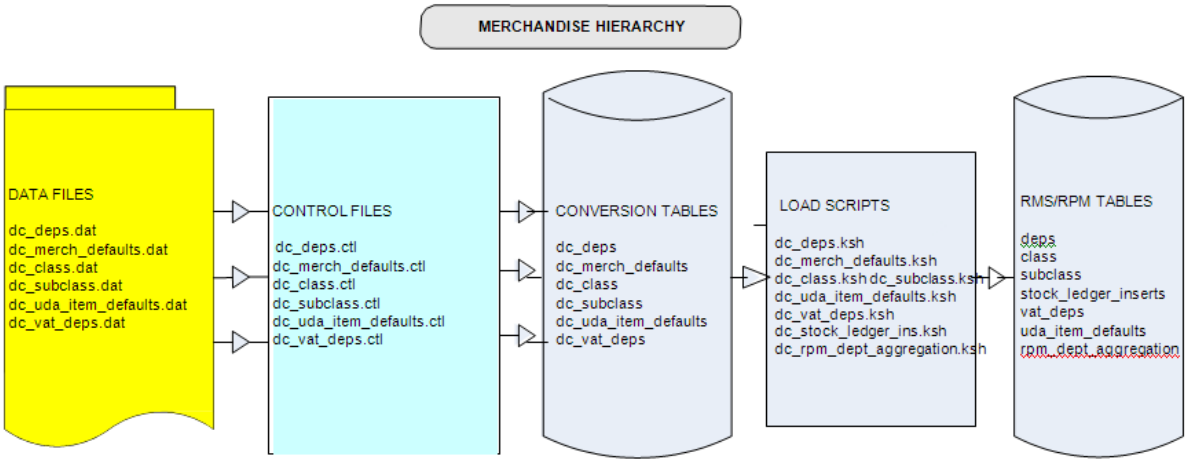
- DEPS
- CLASS
- SUBCLASS
- STOCK_LEDGER_INSERTS
- VAT_DEPS (only if default_tax_type is not GTAX)
- UDA_ITEM_DEFAULTS
- RPM_DEPT_AGGREGATION

The following programs are included in this functional area:

- Load Scripts:
  - dc_merch_defaults.ksh
  - dc_deps.ksh
  - dc_class.ksh
  - dc_subclass.ksh
  - dc_uda_item_defaults.ksh
  - dc_vat_deps.ksh
  - dc_ stock_ledger_ins.ksh
  - dc_rpm_dept_aggregation.ksh
- Control Files:
  - dc_merch_defaults.ctl
  - dc_deps.ctl
  - dc_class.ctl
  - dc_subclass.ctl
  - dc_uda_item_defaults.ctl
  - dc_vat_deps.ctl

Data Flow

The following diagram shows the data flow for the Merchandise Hierarchy functional area:



Data Flow for the Merchandise Hierarchy Functional Area

Prerequisites

Before you begin using the data conversion toolset for Merchandise Hierarchy, you must complete data conversion for the Core functional area.

There are tables that must be loaded manually, because of data dependencies for auto-loading within this functional area. Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following **required** tables must be loaded manually:

- COMPHEAD
- DIVISION
- GROUPS

You must retrieve the assigned data value UDA_VALUES.UDA_VALUE to create the DC_UDA_ITEM_DEFAULT.DAT flat file (see the section, UDA Item Defaults - DC_UDA_ITEM_DEFAULTS Table, in this chapter).

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

Department - DC_DEPS Table

File name: **DC_DEPS.DAT**

Control file: **DC_DEPS.CTL**

Staging table: **DC_DEPS**

This table is used to load the RMS DEPS and the RPM RPM_DEPT_AGGREGATION tables.

Suggested post-loading validation:

- Ensure that DEPS.DEPT is unique.
- Ensure that DEPS.GROUP_NO is a valid GROUPS.GROUP_NO.
- Ensure DEPS.BUYER (if not NULL) is a valid BUYER.BUYER.
- Ensure DEPS.MERCH (if not NULL) is a valid MERCHANT.MERCH.
- Capture the counts from DEPS and RPM_DEPT_AGGREGATION and compare to flat file DC_DEPS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
MERCH_HIER_4	Integer	4	Y	Unique identifier of the merchandise hierarchy level 4.	DEPT	NUMBER(4)
MERCH_HIER_4_DESC	Alpha-numeric	120	Y	Description of the merchandise hierarchy level 4.	DEPT_NAME	VARCHAR2(120)
BUYER	Integer	4	N	Buyer for this merchandise hierarchy level 4. Valid values are from the BUYER field in the BUYER table in RMS.	BUYER	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
MERCHANDISER	Integer	4	N	Merchandiser for this merchandise hierarchy level 4. Valid values are from the MERCH column in the MERCHANT table in RMS.	MERCH	NUMBER(4)
PROFIT_CALC_TYPE	Integer	1	N	Method of accounting for the stock ledger. Valid values are: 1 - Direct cost 2 - Retail inventory If no value is specified in the flat file, the default value is taken from the MERCH_HIER_4_ PROFIT_CALC_TYPE field in the DC_MERCH_ DEFAULTS file.	PROFIT_CALC_TYPE	NUMBER(1)
MERCHANDISE_TYPE	Integer	1	N	Type of merchandise in this merchandise hierarchy level 4. Valid values are: 0 - Normal merchandise 1 - Consignment stock 2 - Concession items If no value is specified in the flat file, the default value is taken from the MERCH_HIER_4_PURCHASE_TYPE field in the DC_MERCH_DEFAULTS file.	PURCHASE_TYPE	NUMBER(1)
MERCH_HIER_3	Integer	4	Y	Identifier of the merchandise hierarchy level 3 of which merchandise hierarchy level 4 is a member. Valid values are in the GROUP_NO field in the GROUPS table in RMS.	GROUP_NO	NUMBER(4)
MERCH_HIER_4_MRKUP_PCT	Numeric	12,4	N	Budgeted intake or markup percentage. The intake percentage, which is the percent of total take that is income, and the markup percent are calculated from this percent base on the value given as the MARKUP_CALC_TYPE. If no value is specified in the flat file, the default value is taken from the MARKUP_PCT field in the DC_MERCH_DEFAULTS file.	MRKUP_PCT	NUMBER(12,4)
TOTAL_MARKET_AMT	Numeric	24,4	N	Total market amount expected for this merchandise hierarchy level 4. Optional.	TOTAL_MARKET_AMT	NUMBER(24,4)

FILE FORMAT					STAGING TABLE DEFINITION	
MARKUP_CALC_TYPE	Alpha-numeric	2	N	Indicates how markup is calculated for this merchandise hierarchy level 4. Valid values are: C - Cost R - Retail If no value is specified in the flat file, the default value is taken from the MERCH_HIER_4_MARKUP_CALC_TYPE field in the DC_MERCH_DEFAULTS file.	MARKUP_CALC_TYPE	VARCHAR2(2)
OTB_CALC_TYPE	Alpha-numeric	1	N	Indicates how open to buy is calculated for this merchandise hierarchy level 4. Valid values are: C - Cost R - Retail If no value is specified in the flat file, the default value is taken from the MERCH_HIER_4_OTB_CALC_TYPE field in the DC_MERCH_DEFAULTS file.	OTB_CALC_TYPE	VARCHAR2(1)
MAX_AVG_COUNTER	Integer	5	N	Maximum count of days with acceptable data to include in projected sales for items within the merchandise hierarchy 4. This provides a way to weigh the existing sales value in the IF_RPM_SMOOTHED_AVG table against new values received. The purpose of this table is to populate projected sales in RPM. If the item has a relatively minimal seasonal swing, this value can be set higher, so that the value will remain stable and take many anomalies to move the value. If the item has a relatively strong seasonal swing, the counter should be set to a lower number, so that the value is more easily moved to reflect a trend or seasonal shift. Required if RPM is used. Default value is 1.	MAX_AVGCOUNTER	NUMBER(5)

FILE FORMAT					STAGING TABLE DEFINITION	
AVG_TOLERANCE_PCT	Numeric	12,4	N	Tolerance percentage used in averaging sales for items. Used to determine if a sales amount received falls within a reasonable range to be considered in the calculation. Values that fall outside the range are considered outliers and are capped at the high or low of the range. This is used by ReSA to populate the IF_RPM_SMOOTHED_AVG table. The purpose of this table is to populate projected sales in RPM. This value sets up a range for appropriate data. The value should be entered as a whole integer; for example, 70 represents 70%. Required if RPM is used. Default value is 1.	AVG_TOLERANCE_PCT	NUMBER(12,4)
VAT_IN_RETAIL	Alpha-numeric	1	N	Indicates whether retail is held with or without VAT. If VAT is not turned on in RMS, then this value should be N. If no value is specified in the flat file, the default value is taken from the MERCH_HIER_4_VAT_IN_RETAIL field in the DC_MERCH_DEFAULTS file.	DEPT_VAT_INCL_IND	VARCHAR2(1)
LOWEST_STRATEGY_LEVEL	Integer	6	N	Lowest level at which a strategy can be defined. Valid values are: 0 - Merchandise hierarchy 4 1 - Merchandise hierarchy 5 2 - Merchandise hierarchy 6 If no value is specified in the flat file, the default value is 0.	LOWEST_STRATEGY_LEVEL	NUMBER(6)
WORKSHEET_LEVEL	Integer	6	N	Indicates the merchandise hierarchy level used to build worksheets for pricing strategies in RPM. This value should be at or above the value in the LOWEST_STRATEGY_LEVEL. Valid values are: 0 - Merchandise hierarchy 4 1 - Merchandise hierarchy 5 2 - Merchandise hierarchy 6 If no value is specified in the flat file, the default value is 0.	WORKSHEET_LEVEL	NUMBER(6)

FILE FORMAT					STAGING TABLE DEFINITION	
HISTORICAL_SALES_PERIOD	Integer	6		Indicates the period used by the merchandise extract to RPM when extracting historical sales.Valid values are: 0 – Week 1 – Month 2 - Half year 3 – Year If no value is specified in the flat file, the default value is 0.	HISTORICAL_SALES_LEVEL	NUMBER(6)
REGULAR_SALES_IND	Integer	6	N	Indicates whether regular price sales are included as part of the historical sales extracted by the merchandise extract to RPM. Valid values are: 0 - Do not include 1 – Include If no value is specified in the flat file, the default value is 0.	REGULAR_SALES_IND	NUMBER(6)
CLEARANCE_SALES_IND	Integer	6	N	Indicates whether clearance price sales are included as part of the historical sales extracted by the merchandise extract to RPM. Valid values are: 0 - Do not include 1 – Include If no value is specified in the flat file, the default value is 0.	CLEARANCE_SALES_IND	NUMBER(6)
PROMOTIONAL_SALES_IND	Integer	6	N	Indicates whether promotional price sales are included as part of the historical sales extracted by the merchandise extract to RPM. Valid values are: 0 - Do not include 1 – Include If no value is specified in the flat file, the default value is 0.	PROMOTIONAL_SALES_IND	NUMBER(6)
INCLUDE_WH_ON_HAND	Integer	6	N	Indicator used by the merchandise extract to RPM to determine whether to include the warehouse on hand quantity when calculating sell thru and price change impact. If no value is specified in the flat file, the default value is 0.	INCLUDE_WH_ON_HAND	NUMBER(6)

FILE FORMAT					STAGING TABLE DEFINITION	
INCLUDE_WH_ON_ORDER	Integer	6	N	Indicator used by the merchandise extract to RPM to determine whether to include the warehouse on order quantity when calculating the total on order quantity. If no value is specified in the flat file, the default value is 0.	INCLUDE_WH_ON_ORDER	NUMBER(6)
PRICE_CHANGE_AMOUNT_CALC_TYPE	Integer	6	N	Calculation method for the price change amount column on the Worksheet and Worksheet status screens. Valid values are: 0 - Current-New 1 - New-Current If no value is specified in the flat file, the default value is 0.	PRICE_CHANGE_AMOUNT_CALC_TYPE	NUMBER(6)
RETAIL_CHG_HIGHLIGHT_DAYS	Integer	4	N	Defines a window of recent price changes. The worksheet will highlight past price changes that fall within this window of time.	RETAIL_CHG_HIGHLIGHT_DAYS	NUMBER(4)
COST_CHG_HIGHLIGHT_DAYS	Integer	4	N	Defines a window of recent cost changes. The worksheet highlights past cost changes that fall within this window of time.	COST_CHG_HIGHLIGHT_DAYS	NUMBER(4)
PEND_COST_CHG_WINDOW_DAYS	Integer	4	N	Indicates how many days forward the worksheet looks to find upcoming cost changes.	PEND_COST_CHG_WINDOW_DAYS	NUMBER(4)
PEND_COST_CHG_HIGHLIGHT_DAYS	Integer	4	N	Defines a window of upcoming cost changes. The worksheet highlights upcoming cost changes that fall within this window of time.	PEND_COST_CHG_HIGHLIGHT_DAYS	NUMBER(4)

Merchandise Hierarchy Defaults - DC_MERCH_DEFAULTS Table

File name: **DC_MERCH_DEFAULTS.DAT**

Control file: **DC_MERCH_DEFAULTS.CTL**

Staging table: **DC_MERCH_DEFAULTS**

The purpose of this table is a place to indicate more system-wide defaults.

DEFAULT_ALL_MERCH_HIER_5 and 6 are used to default class or subclass values (create only one class or subclass per department or class). If DEFAULT_CLASS is Y, only one class and subclass is inserted per department. If DEFAULT_SUBCLASS is Y, one subclass is inserted for each class. If defaulting is used, the corresponding DC_SUBCLASS.DAT and DC_CLASS.DAT (if applicable) files are assumed to be empty.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
Default_merch_heir_5	Alpha-numeric	1	Y	Indicates whether to default the merchandise hierarchy levels 5 and 6 (RMS class and subclass) records based on each merchandise hierarchy level 4 (RMS department). Valid values are Y and N.	DEFAULT_CLASS	VARCHAR2(1)
Default_merch_heir_6	Alpha-numeric	1	Y	Indicates whether to default the merchandise hierarchy level 6 (RMS subclass) records. If DEFAULT_MERCH_HIER_5 (RMS class) is Yes, then it is assumed that MERCH_HIER_6 values are defaulted as well. Valid values are Y and N.	DEFAULT_SUBCLASS	VARCHAR2(1)
Merch_heir_4_profit_calc_type	Integer	1	Y	Default value to be used for all MERCH_HIER_4 records that do not have a profit calc type value specified. Valid values are: 1 - Direct cost 2 - Retail inventory	DEPT_PROFIT_CALC_TYPE	NUMBER
Merch_heir_4_purchase_type	Integer	1	N	Purchase type default value for MERCH_HIER_4 records that do not have a merchandise type value specified. Valid values are: 0 - Normal merchandise 1 - Consignment stock 2 - Concession items	DEPT_PURCHASE_TYPE	NUMBER
Merch_heir_4_MRKUP_PCT	Integer	12,4	Y	Indicates whether the field specifies the intake or markup is determined by the value of the DC_DEPS. MARKUP_CALC_TYPE field. A value of R indicates that this field specifies the budgeted intake, which is synonymous with markup percent of retail. A value of C indicates that this field specifies the budgeted markup, which is synonymous with markup percent of cost. If no value is specified in the flat file, the default value is taken from the MARKUP_PCT field in the DC_MERCH_DEFAULTS file.	DEPT_MRKUP_PCT	NUMBER

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
Merch_heir_4_markup_calc_type	Alpha-numeric	2	Y	Indicates whether the markup calculation type should be Cost or Retail for MERCH_HIER_4 records. Valid values are: C – Cost R – Retail	DEPT_MARKUP_CALC_TYPE	VARCHAR2
Merch_heir_4_otb_calc_type	Alpha-numeric	1	Y	Indicates whether the open to buy (OTB) calculation type should be Cost or Retail for MERCH_HIER_4 records. Valid values are: C – Cost R – Retail	DEPT_OTB_CALC_TYPE	VARCHAR2
Merch_hier_4_VAT_IN_RETAIL	Alpha-numeric	1	Y	Indicates whether retail is held with VAT in the MERCH_HIER_4 level. If VAT is not turned on in RMS, this value should be N.	DEPT_VAT_INCL_IND	VARCHAR2(1)
Merch_hier_5_VAT_IN_RETAIL	Alpha-numeric	1	Y	Indicates whether retail is held with VAT in the MERCH_HIER_5 level. If VAT is not turned on in RMS, this value should be N.	CLASS_VAT_INCL_IND	VARCHAR2(1)

Class - DC_CLASS Table

File name: **DC_CLASS.DAT**

Control file: **DC_CLASS.CTL**

Staging table: **DC_CLASS**

Suggested post-loading validation:

- Ensure that the CLASS.DEPT/CLASS.CLASS combination is unique.
- Ensure that CLASS.DEPT is a valid DEPS.DEPT.
- Capture the count from CLASS and compare to flat file DC_CLASS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 of which merchandise hierarchy level 5 is a member. Valid values are in the DEPT field in the DEPS table.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Unique identifier of the merchandise hierarchy level 5.	CLASS	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
MERCH_HIER_5_NAME	Alpha-numeric	120	Y	Description of the merchandise hierarchy level 5.	CLASS_NAME	VARCHAR2(120)
VAT_IN_RETAIL	Alpha-numeric	1	N	Indicates whether retail is held with VAT. If VAT is not turned on in RMS, this value should be N. If no value is specified in the flat file, the value is defaulted from the MERCH_HIER_5_VAT_IN_RETAIL field in the DC_MERCH_DEFAULTS file.	CLASS_VAT_IND	VARCHAR2(1)

Subclass - DC_SUBCLASS Table

File name: **DC_SUBCLASS.DAT**

Control file: **DC_SUBCLASS.CTL**

Staging table: **DC_SUBCLASS**

Suggested post-loading validation:

- Ensure that the SUBCLASS.DEPT/SUBCLASS.CLASS/SUBCLASS.SUBCLASS combination is unique.
- Ensure that the SUBCLASS..DEPT/SUBCLASS.CLASS combination exists in CLASS.
- Capture the count from SUBCLASS and compare to flat file DC_SUBCLASS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 of which merchandise hierarchy level 6 is a member. Valid values are in the DEPT field in the DEPS table.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifier of the merchandise hierarchy level 5 of which merchandise hierarchy level 6 is a member. Valid values are in the CLASS field in the CLASS table.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Unique identifier of the merchandise hierarchy level 6.	SUBCLASS	NUMBER(4)
MERCH_HIER_6_NAME	Alpha-numeric	120	Y	Description of the merchandise hierarchy level 6.	SUBCLASS_NAME	VARCHAR2(120)

VAT Departments - DC_VAT_DEPS Table

File name: **DC_VAT_DEPS.DAT**

Control file: **DC_VAT_DEPS.SQL**

Staging table: **DC_VAT_DEPS**

Suggested post-loading validation:

- Ensure that every VAT_DEPS.VAT_REGION/VAT_DEPS.DEPT/ VAT_DEPS.VAT_TYPE combination is unique.
- Ensure that VAT_DEPS.VAT_REGION is a valid VAT_REGION.VAT_REGION.
- Ensure VAT_DEPS.DEPT is a valid DEPS.DEPT.
- Ensure VAT_DEPS.VAT_CODE is a valid VAT_CODES.VAT_CODE.
- Capture the count from VAT.DEPS and compare to flat file DC_VAT_DEPS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
Merch_heir_4	Integer	4	Y	Unique identifier of the merch hierarchy level 4.	DEPT	NUMBER(4)
VAT_REGION	Integer	4	Y*	Unique identifier of VAT region. VAT region is determined by the VAT authority. This value should to a value in the DC_VAT_REGION.DAT file.	VAT_REGION	NUMBER(4)
VAT_TYPE	Alpha-numeric	1	Y*	Indicates whether the VAT rate is used for purchasing (Cost), selling (Retail), or both. Valid values are from the VTTP code type: C, R, or B.	VAT_TYPE	VARCHAR2(1)
VAT_CODE	Alpha-numeric	6	Y*	Unique identifier of VAT code. Valid values include: S= - StandardZ= - Zero. This value should correspond to a value from the DC_VAT_CODES DAT file.	VAT_CODE	VARCHAR2(6)
REVERSE_VAT_IND	Alpha-numeric	1	Y	Indicates if the department is subjected to reverse charge VAT. Valid values are 'Y' or 'N'.	REVERSE_VAT_IND	VARCHAR2(1)

Note: The asterisk in the table above indicates that the field is required when VAT is turned on in RMS and default_tax_type is not GTAX.

UDA Item Defaults - DC_UDA_ITEM_DEFAULTS Table

File name: **DC_UDA_ITEM_DEFAULTS.DAT**

Control file: **DC_UDA_DEFAULTS.CTL**

Staging table: **DC_UDA_ITEM_DEFAULTS**

Suggested post-loading validation (sequence after dc_load_merch.ksh):

- Ensure that the UDA_ITEM_DEFAULTS.UDA_ID/UDA_ITEM_DEFAULTS.SEQ_NO combination is unique.
- Ensure that UDA_ITEM_DEFAULTS.UDA_ID is a valid UDA.UDA_ID.
- Ensure that UDA_ITEM_DEFAULTS.DEPT is a valid DEPS.DEPT.
- Ensure that UDA_ITEM_DEFAULTS.DEPT/UDA_ITEM_DEFAULTS.CLASS combination exists on CLASS (if UDA_ITEM_DEFAULTS.CLASS is not NULL).
- Ensure that UDA_ITEM_DEFAULTS.DEPT/UDA_ITEM_DEFAULTS.CLASS/ UDA_ITEM_DEFAULTS.SUBCLASS combination exists on SUBCLASS (if UDA_ITEM_DEFAULTS.SUBCLASS is not NULL).
- Ensure that UDA_ITEM_DEFAULTS.UDA_ID/ UDA_ITEM_DEFAULTS.UDA_VALUE combination exists in UDA_VALUES (if UDA_ITEM_DEFAULTS.UDA_VALUE is not NULL).
- Capture the count from UDA_ITEM_DEFAULTS and compare to flat file DC_UDA_ITEM_DEFAULTS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
UDA_ID	Integer	5	Y	Unique identifier of the user - defined attributes (UDA) to be defaulted. Valid values are in the UDA_ID field in the UDA table.	UDA_ID	NUMBER(5)
MERCH_HIER_4	Integer	4	Y	Merchandise hierarchy level 4 associated with the defaulted UDA.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	N	Merchandise hierarchy level 5 associated with the defaulted UDA. Optional, but required if the MERCH_HIER_6 field is populated.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	N	Merchandise hierarchy level 6 associated to the defaulted UDA. Optional.	SUBCLASS	NUMBER(4)
UDA_VALUE	Integer	3	Y	Only the UDA_ID DISPLAY_TYPE of LV is defaulted to the hierarchy specified; therefore, UDA_VALUE is required. Valid values are in the UDA_VALUE field in the UDA_VALUES table for the UDA_ID specified.	UDA_VALUE	NUMBER(3)

Note: If any records are in the BAD or DISCARD file, the RMS table must be truncated and the entire file must be rerun.

Load Scripts

DC_MERCH_DEFAULTS.KSH

This ksh script will be called to call SQLLOADER to load flat file data to staging table.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_MERCH_DEFAULTS staging table.

Required file to load: dc_merch_defaults.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_DEPS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_DEPS staging table.

LOAD_DEPS– This function loads the RMS DEPS table. The required files for this function dc_merch_defaults.dat and dc_deps.dat. The dc_merch_defaults values should be fetched into a rowtype local variable. Use the variables as specified in the below default table to load the RMS DEPS table from DC_DEPS. If the dc_merch_defaults.default_class_ind is 'Y' then only one class and subclass will be inserted for each dept. Use the DC_DEPS and DC_MERCH_DEFAULT to insert into the RMS class and subclass tables. The id and name will be the same as dept for all classes and subclasses. In the CLASS table, CLASS_ID column will be populated with class_id_sequence value. In the SUBCLASS table, SUBCLASS_ID column will be populated with subclass_id_sequence value.

DC_DEPS to DEPS Column Defaults

Field Name (RMS Table)	Default Value	Comments
PROFIT_CALC_TYPE	DC_MERCH_DEFAULTS DEPT_PROFIT_CALC_TYPE	If DC_DEPS.PROFIT_CALC_TYPE is NULL, use the value from the MERCH_DEFAULTS table.
PURCHASE_TYPE	NVL(SYSTEM.DEFAULTS DEPT_PURCHASE_TYPE, 0)	If DC_DEPS.PURCHASE_TYPE is NULL, use the value from the MERCH_DEFAULTS table. If that is NULL, then default to 0.
BUD_INT	DC_MERCH_DEFAULTS DEPT_BUD_INT	If DC_DEPS.MARKUP_CALC_TYPE is R, use DC_DEPS.MRKUP_PCT to populate the DEPS.BUD_INT RMS field. If DC_DEPS.MRKUP_PCT is NULL, use MERCH_DEFAULTS. DEPT_MRKUP_PCT to populate DEPS.BUD_INT. If DC_DEPS.MARKUP_CALC_TYPE is C, populate the DEPS.BUD_INT RMS field using the following equation: `BUD_INT = round(DC_DEPS.MRKUP_PCT * 100/DC_DEPS.MRKUP_PCT + 100),4).`
BUD_MKUP	DC_MERCH_DEFAULTS DEPT_BUD_MKUP	If DC_DEPS.MARKUP_CALC_TYPE is C, use DC_DEPS.MRKUP_PCT to populate the DEPS.BUD_MKUP RMS field. If DC_DEPS.MRKUP_PCT is NULL, use MERCH_DEFAULTS. DEPT_MRKUP_PCT to populate DEPS.BUD_MKUP. If DC_DEPS.MARKUP_CALC_TYPE is R, populate the DEPS.BUD_MKUP RMS field using the following equation: `BUD_MKUP = round(DC_DEPS.MRKUP_PCT * 100/(100 – DC_DEPS.MRKUP_PCT),4).`
MARKUP_CALC_TYPE	DC_MERCH_DEFAULTS DEPT_MARKUP_CALC_TYPE	If DC_DEPS.MARKUP_CALC_TYPE is NULL, use the value from the MERCH_DEFAULTS table.
OTB_CALC_TYPE	DC_MERCH_DEFAULTS DEPT_OTB_CALC_TYPE	If DC_DEPS.OTB_CALC_TYPE is NULL, use the value from the MERCH_DEFAULTS table.
DEPT_VAT_INCL_IND	DC_MERCH_DEFAULTS DEPT_VAT_INCL_IND	If DC_DEPS.DEPT_VAT_INCL_IND is NULL, use the value from the MERCH_DEFAULTS table.

Required file to load: dc_merch_defaults.dat, dc_deps.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_CLASS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_CLASS staging table.

LOAD_CLASS– This function contains a PL/SQL block that selects from the DC_CLASS staging table and inserts the data to the RMS CLASS and possibly SUBCLASS tables.

> **Note**: This load will not run if the subclasses are defaulted in the LOAD_DEPS table (that is, no dc_class.dat file exists).
>
> The script first gets the indicators from the DC_MERCH_ DEFAULTS table. If the DEFAULT_CLASS indicator is not set to Y, the records from DC_CLASS are loaded into the RMS CLASS table. If the DEFAULT_SUBCLASS indicator is set to Y, only one subclass is inserted for each class. The subclass ID defaults to the class ID value.

The following table defines the default value in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_CLASS to CLASS Column Defaults

Field Name (RMS Table)	Default Value	Comments
CLASS_VAT_INCL_IND	SYSTEM_DEFAULTS.CLASS_ VAT_INCL_IND	If DC_CLASS. CLASS_VAT_INCL_IND is NULL, use the value from MERCH_DEFAULTS.

Required file to load: dc_merch_defaults.dat, dc_class.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_SUBCLASS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_SUBCLASS staging table.

LOAD_SUBCLASS– This function contains a PL/SQL block that selects from the DC_SUBCLASS staging table and inserts the data to the RMS SUBCLASS.

> **Note**: This load will not be run if the subclasses are defaulted in the LOAD_DEPS or LOAD_CLASSES functions (that is, no dc_subclass.dat file). The script first gets the indicators from the DC_MERCH_ DEFAULTS table. If the DEFAULT_SUBCLASS indicator is not set to Y, the function selects from DC_SUBCLASS and inserts into the RMS SUBCLASS table.

Required file to load: dc_merch_defaults.dat, dc_subclass.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_STOCK_LEDGER_INS.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_TSF_ENTITY– This function creates records in the RMS STOCK_LEDGER_INSERTS table for every new department and subclass loaded. The function performs an insert/select from the DC_DEPS and DC_SUBCLASS tables to insert the appropriate information (with type_code D or B, respectively) into the STOCK_LEDGER_INSERTS table.

Required file to load: dc_deps.dat, dc_subclass.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_VAT_DEPS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_VAT_DEPS staging table.

LOAD_VAT_DEPS– This function selects from the DC_VAT_DEPS table and inserts the records into RMS VAT_DEPS if the system options vat_ind is equal to Y and default tax type is NOT 'GTAX' (i.e. 'SVAT' is used). All the columns from the staging oracle table defined above will directly map to the RMS table.

Required file to load: dc_vat_deps.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UAD_ITEM_DEFAULTS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA_ITEM_DEFAULTS staging table.

LOAD_UDA_ITEM_DEFAULTS– This function contains a PL/SQL block that selects from the DC_UDA_ITEM_DEFAULTS staging table and inserts the data to the RMS UDA_ITEM_DEFAULTS table. All the columns from the staging table defined above map directly to the RMS table. The following table defines the default value in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_UDA_ITEM_DEFAULTS to UDA_ITEM_DEFAULTS Column Defaults

Field Name (RMS Table)	Default Value	Comments
SEQ_NO	SEQ_NO + 1	Based on dept(class(subclass)). Use analytic function.
HIERARACHY_VALUE	1,2 OR 3	If subclass is not NULL then 3; if class is not NULL then 2; if dept is not NULL
REQUIRED_IND	N	Value Defaults to N if the file value is empty

Required file to load: dc_uda_item_defaults.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_RPM_DEPT_AGGREGATION.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_RPM_DEPT_AGGREGATION – This function selects data from the DC_DEPS table and inserts the records into the RPM_DEPT_AGGREGATION table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_DEPS to RPM_DEPT_AGGREGATION Column Defaults

Field Name (RMS Table)	Default Value	Comments
DEPT_AGGREGATION_ID	Generated SEQ_NO	
LOWEST_STRATEGY_LEVEL	0	Value defaults to 0 if the file value is empty.
WORKSHEET_LEVEL	0	Value defaults to 0 if the file value is empty.
HISTORICAL_SALES_LEVEL	0	Value defaults to 0 if the file value is empty.
REGULAR_SALES_IND	0	Value defaults to 0 if the file value is empty.
CLEARANCE_SALES_IND	0	Value defaults to 0 if the file value is empty.
PROMOTIONAL_SALES_IND	0	Value defaults to 0 if the file value is empty.
INCLUDE_WH_ON_HAND	0	Value defaults to 0 if the file value is empty.
INCLUDE_WH_ON_ORDER	0	Value defaults to 0 if the file value is empty.
PRICE_CHANGE_AMOUNT_CALC_TYPE	0	Value defaults to 0 if the file value is empty.

Required file to load: dc_deps.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Post-Loading Requirements

After using the data conversion toolset for this functional area, there are additional tables that must be loaded manually before you proceed with data conversion for subsequent functional areas, because of data dependencies.

Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following are required tables that require manual data loading:

- RPM_MERCH_RETAIL_DEF
- HIERARCHY_PERMISSION (Retail Security Manager [RSM] table)

Additionally, all department UDA defaults must be set up manually where UDA_ITEM_DEFAULTS.REQUIRED_IND = Y.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts are executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_deps.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

5

Organizational Hierarchy

This chapter describes the organization hierarchy data conversion. Data must be loaded in the following order:

- Warehouse
- Store

Prerequisites

Before you begin using the data conversion toolset for Organizational Hierarchy, you must complete data conversion for the following functional areas:

- Core
- Merchandise Hierarchy

There are tables that must be loaded manually, because of data dependencies for auto-loading within this functional area. Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following **required** tables must be loaded manually:

- CHAIN
- AREA

Warehouse Overview

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- ADDR
- WH
- WH_ADD
- STOCK_LEDGER_INSERTS
- TRANSIT_TIMES (applicable to both store and warehouses)
- COST_ZONE
- COST_ZONE_GROUP_LOC

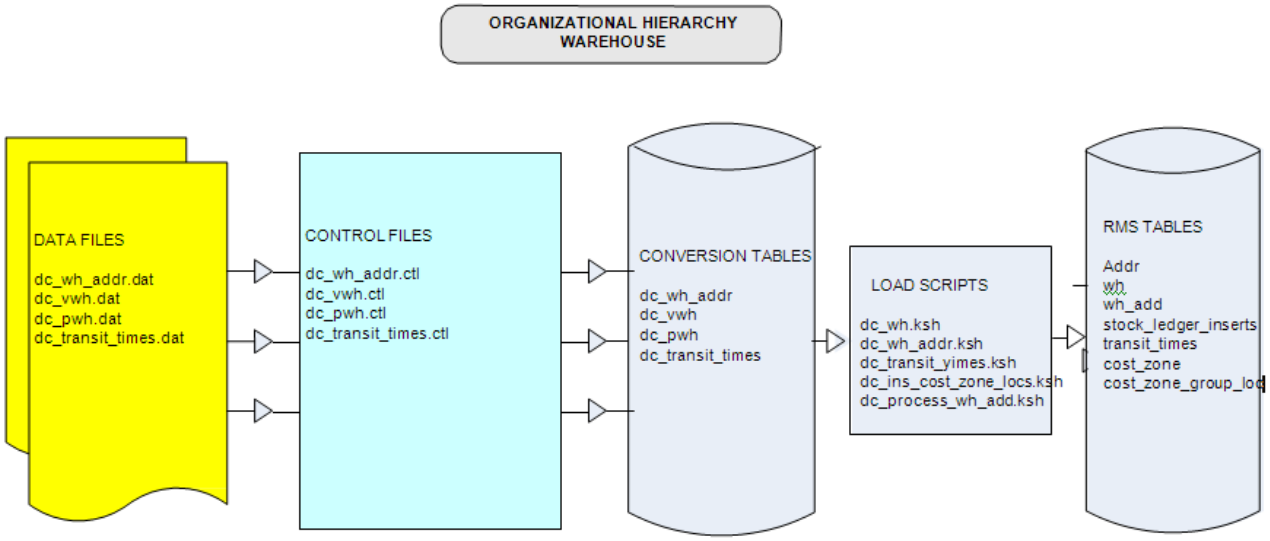
The following programs are included in this functional area:

- Load scripts:
  - dc_wh.ksh
  - dc_wh_addr.ksh
  - dc_transit_times.ksh
  - dc_ins_cost_zone_locs.ksh
  - dc_process_wh_add.ksh
- Control files:
  - dc_wh.ksh
  - dc_pwh.ctl
  - dc_vwh.ctl

- dc_wh_addr.ctl
- dc_transit_times.ctl

Data Flow

The following diagram shows the data flow for the Organizational Hierarchy Warehouse functional area:



Data Flow for the Organizational Hierarchy Warehouse Functional Area

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must create in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the Staging Table Definition columns Field Name and Data Type (including length) define the physical staging table.

DC_WH_ADDR Table

File name: **DC_WH_ADDR.DAT**

Control file: **DC_WH_ADDR.CTL**

Staging table: **DC_WH_ADDR**

Suggested post-loading validation:

- Ensure that ADDR.STATE is a valid STATE.STATE.
- Ensure that ADDR.COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.
- Capture counts from ADDR where ADDR.MODULE = WH and compare to flat file DC_WH_ADDR.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WAREHOUSE_ID	Alpha-numeric	20	Y	Primary identifier for the warehouse for which the address record applies. In a multi-channel environment, only the physical warehouse should contain address records, so this field will contain physical warehouse IDs.	KEY_VALUE_1	VARCHAR2(20)
ADDR_TYPE	Alpha-numeric	2	Y	Type of address for this warehouse. Valid values are: 01 – Business 02 – Postal 03 – Returns 04 – Order 05 – Invoice 06 – Remittance Additional address types can be defined in the RMS ADD_TYPE table. The required address types for a warehouse are definable in the RMS ADD_TYPE_MODULE table, where MODULE=WH.	ADDR_TYPE	VARCHAR2(2)
PRIMARY_ADDR_IND	Alpha-numeric	1	Y	Indicates whether this address is the primary for the warehouse and address type. Valid values are Y (primary) and N (non-primary).	PRIMARY_ADDR_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CONTACT_NAME	Alpha-numeric	120	N	Name of the primary contact person at this warehouse address.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	N	Phone number of the contact person at this warehouse address.	CONTACT_PHONE	VARCHAR2(20)
CONTACT_FAX	Alpha-numeric	20	N	Fax number of this warehouse address.	CONTACT_FAX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	E-mail address of the contact person at this warehouse address.	CONTACT_EMAIL	VARCHAR2(100)
CONTACT_TELEX	Alpha-numeric	20	N	Telex number of the contact person at this warehouse address.	CONTACT_TELEX	VARCHAR2(20)
ADDR_LINE_1	Alpha-numeric	240	Y	First line of the address of this warehouse and address type.	ADD_1	VARCHAR2(240)
ADDR_LINE_2	Alpha-numeric	240	N	Second line of the address of this warehouse and address type.	ADD_2	VARCHAR2(240)
ADDR_LINE_3	Alpha-numeric	240	N	Third line of the address of this warehouse and address type.	ADD_3	VARCHAR2(240)
CITY	Alpha-numeric	120	Y	City of this address of the warehouse and address type.	CITY	VARCHAR2(120)
COUNTY	Alpha-numeric	250	N	County of the address of this warehouse and address type.	COUNTY	VARCHAR2(250)
STATE	Alpha-numeric	3	N	State of the address of this warehouse and address type. Values in this column must exist in the RMS STATE table.	STATE	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	Postal code (for example, ZIP code) of the address for this warehouse and address type.	POST	VARCHAR2(30)
COUNTRY_ID	Alpha-numeric	3	Y	Country code of the address of this warehouse and address type. Values in this column must exist in the RMS COUNTRIES table.	COUNTRY_ID	VARCHAR2(3)
JURISDICTION_CODE	Alpha-numeric	10	N	Jurisdiction code for the address	JURISDICTION_CODE	VARCHAR2(10)

DC_PWH Table

File name: **DC_PWH.DAT**

Control file: **DC_PWH.CTL**

Staging table: **DC_PWH**

Suggested post-loading validation:

- Ensure that WH.WH is unique.
- If WH.ORG_HIER_TYPE has a value of 1, ensure that WH.ORG_HIER_VALUE is a valid COMPHEAD.COMPANY.
- If WH.ORG_HIER_TYPE has a value of 10, ensure that WH.ORG_HIER_VALUE is a valid CHAIN.CHAIN.
- If WH.ORG_HIER_TYPE has a value of 20, ensure that WH.ORG_HIER_VALUE is a valid AREA.AREA.
- If WH.ORG_HIER_TYPE has a value of 30, ensure that WH.ORG_HIER_VALUE is a valid REGION.REGION.
- If WH.ORG_HIER_TYPE has a value of 40, ensure that WH.ORG_HIER_VALUE is a valid DISTRICT.DISTRICT.
- If WH.ORG_HIER_TYPE has a value of 50, ensure that WH.ORG_HIER_VALUE is a valid STORE.STORE.
- Ensure that WH.VAT_REGION is a valid VAT_REGION.VAT_REGION, if WH.STOCKHOLIDNG_IND = Y.
- Ensure that WH.CURRENCY_CODE is a valid CURRENCIES.CURRENCY_CODE.
- Ensure that WH.ORG_UNIT_ID (if not NULL) is a valid ORG_UNIT.ORG_UNIT_ID.
- Ensure that WH.TSF_ENTITY_ID is a valid TSF_ENTITYT.TSF_ENTITY_ID if WH.STOCKHOLIDNG_IND = Y.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WAREHOUSE_ID	Integer	10	Y	Unique identifier of the warehouse being defined. In a multi-channel environment, this contains the physical warehouse ID. In a single-channel environment, there is no distinction between physical and virtual warehouses. This value must be unique across all warehouses (physical and virtual) and stores.	WH	NUMBER(10)
WAREHOUSE_NAME	Alpha-numeric	150	Y	Name of the warehouse being defined.	WH_NAME	VARCHAR2(150)

FILE FORMAT					STAGING TABLE DEFINITION	
PRIMARY_ VIRTUAL_ WH_ID	Integer	10	N	(Applicable only in a multi-channel environment. Required in a multi-channel environment). Identifier of the primary virtual warehouse within this physical warehouse. The value must be a valid virtual warehouse loaded in the VWH file that exists within this physical warehouse. The primary VWH is used throughout RMS in various transactions for which only a physical warehouse has been specified.	PRIMARY_ VWH	NUMBER(10)
CURRENCY_ CODE	Alpha-numeric	3	Y	Currency in which all cost and retail values for this warehouse are represented. Valid values must exist in the RMS CURRENCIES table.	CURRENCY _CODE	VARCHAR2(3)
BREAK_PACK_ IND	Alpha-numeric	1	N	Indicates whether this warehouse is capable of distributing less than the supplier case quantity (supplier pack size). Valid values are Y and N.	BREAK_ PACK_ IND	VARCHAR2(1)
REDIST_WH_ IND	Alpha-numeric	1	N	Indicates whether this warehouse is considered a redistribution warehouse, which is a dummy warehouse usable for creating purchase orders in advance of knowing the final order to locations. This flag is only used by the RMS Order Redistribution report, as a query criterion for displaying POs that require redistribution to the final locations. Valid values are Y and N.	REDIST_W H_IND	VARCHAR2(1)
DELIVERY_ POLICY	Alpha-numeric	6	Y	Warehouse delivery policy for shipments from the warehouse. Valid values are: NEXT - Next day NDD - Next delivery day NEXT indicates that deliveries are made on the next warehouse open day. NDD indicates that deliveries are made only on scheduled days.	DELIVERY_ POLICY	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
FORECAST_WH_IND	Alpha-numeric	1	N	Indicates whether the warehouse can be forecasted. Value are Y and N. Only warehouses with a value of Y are visible to the forecasting tool (RDF). In a multi-channel environment, this parameter only needs to be defined for virtual warehouses, so that it can be passed as NULL for the physical warehouse.	FORECAST_WH_IND	VARCHAR2(1)
REPL_IND	Alpha-numeric	1	N	Indicates whether this warehouse can be used to replenish other locations. Valid values are Y and N. Y indicates that inventory from this warehouse can be used to replenish other locations. In a multi-channel environment, this parameter only needs to be defined for virtual warehouses, so that it can be passed as NULL for the physical warehouse.	REPL_IND	VARCHAR2(1)
REPL_WH_LINK	Integer	10	N	Replenishable warehouse that is linked to this virtual warehouse. This link implies that the virtual warehouse is included in the net inventory calculations for the replenishable warehouse. This field is should only be definable in a single-channel environment and where the value in the repl_ind field is N.	REPL_WH_IND	NUMBER(10)
IB_IND	Alpha-numeric	1	N	Indicates whether the warehouse is an investment buy warehouse.	IB_IND	VARCHAR2(1)
IB_WH_LINK	Integer	10	N	Investment buy warehouse that is linked to the virtual warehouse. This link implies that the virtual warehouse is included in the net inventory calculations for the investment buy warehouse. This field should only contain a value when the IB_IND is N.	IB_WH_LINK	NUMBER(10)

FILE FORMAT					STAGING TABLE DEFINITION	
AUTO_IB_CLEAR	Alpha-numeric	1	N	Indicates whether the investment buy inventory should be automatically transferred to the turn (replenishable) warehouse when an order is received by the turn warehouse. Valid values are Y and N.	AUTO_IB_CLEAR	VARCHAR2(1)
INBOUND_HANDLING_DAYS	Integer	2	Y	Number of days that the warehouse requires to receive any item and get it to the shelf so that it is ready to pick. Valid value is a number from 0 to 99.	INBOUND_HANDLING_DAYS	NUMBER(2)
WH_NAME_SECONDARY	Alpha-numeric	150	N	Secondary description of the warehouse. This value is used to support multi-language, where the primary description may contain characters not easily sortable.	WH_NAME_SECONDARY	VARCHAR2(150)
EMAIL	Alpha-numeric	100	N	Primary e-mail for the warehouse.	EMAIL	VARCHAR2(100)
VAT_REGION	Integer	4	N	Required when VAT_IND in SYSTEM_OPTIONS is Y. Contains the warehouse VAT region, used by RMS to determine the VAT rates applicable at this location.	VAT_REGION	NUMBER(4)
ORG_HIER_TYPE	Integer	4	N	For reporting purposes, this field, along with the ORG_HIER_VALUE field, can be used to define a level and value of the organizational hierarchy with which this warehouse is associated. This field defines the level of the organizational hierarchy defined in the ORG_HIER_VALUE field. Valid values are: 1 – Company 10 – Chain 20 – Area 30 – Region 40 – District 50 – Store	ORG_HIER_TYPE	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
ORG_HIER_ VALUE	Integer	10	N	(See ORG_HIER_TYPE description). ID of the organizational hierarchy value as defined by the ORG_HIER_TYPE. For example, if the ORG_HIER_TYPE is 20 (area), this field should contain a valid area ID.	ORG_HIER_ VALUE	NUMBER(10)
DUNS_LOC	Alpha-numeric	4	N	Dun and Bradstreet number used to identify the warehouse location. This is reference-only data.	DUNS_LOC	VARCHAR2(4)
DUNS_ NUMBER	Alpha-numeric	9	N	Dun and Bradstreet number used to identify the warehouse. This is reference-only data.	DUNS_ NUMBER	VARCHAR2(9)
ORG_UNIT_ID	Numeric	15	N	Unique identifier for the Oracle Organizational ID.	ORG_UNIT_ ID	NUMBER(15)

DC_VWH Table

File name: **DC_VWH.DAT**

This VWH.DAT file contains the virtual warehouse locations details for each physical warehouse. This file is to be created and loaded into RMS only when multi-channel functionality is enabled (SYSTEM_OPTIONS. MULTICHANNEL_IND = Y). Otherwise, this file is not necessary, and only the DC_PWH.DAT file is required.

Control file: **DC_VWH.CTL**

Staging table: **DC_VWH**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
VIRTUAL_ WH_ID	Integer	10	Y	Unique identifier for the virtual warehouse. This value must be unique across all warehouses (physical and virtual) and stores.	WH	NUMBER(10)
VIRTUAL_WH_ NAME	Alpha-numeric	150	Y	Name for the virtual warehouse being defined.	WH_NAME	VARCHAR2 (150)
PHYSICAL_ WAREHOUSE_ ID	Integer	10	Y	ID of the physical warehouse in which this virtual warehouse resides. To be valid, the physical warehouse must already exist in RMS and be loaded separately from the physical warehouse file.	PHYSICAL_WH	NUMBER(10)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
RESTRICTED_IND	Alpha-numeric	1	N	Indicator used to restrict virtual warehouses from receiving stock during an inbound type transaction (such as positive SOH inventory adjustment, PO over-receipt), when stock needs to be prorated across virtual warehouses within a physical warehouse, because a virtual warehouse in the physical warehouse has not been identified for the transaction. The indicator restricts the virtual warehouse from receiving stock unless all valid virtual warehouses determined by the system are restricted; then the stocks are distributed across those restricted virtual warehouses. This indicator is used only in a multi-channel environment. It is always set to 'N' in a single-channel environment.	RESTRICTED_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
TECTED_ IND	Alpha-numeric	1	N	Indicates whether the virtual warehouse is affected last in transactions where inventory is removed, or affected first in short-shipment type transactions where inventory is being added. The indicator is used in any outbound or inventory removal type transactions (such as returns to vendor [RTV], negative stock on hand [SOH] inventory adjustments), when the system has to distribute the transaction quantity across virtual warehouses within a physical warehouse for one of these reasons: ▪ A virtual warehouse has not been specified or could not be derived. ▪ A virtual warehouse does not have enough stock to cover the transaction quantity, and stock needs to be pulled from other virtual warehouses within the physical warehouse. The indicator is also used for inbound type transactions where there is some sort of short-shipment (for example, a short-shipment for a PO). The indicator determines which virtual warehouses have their order quantity fulfilled first with the receipt quantity. Note that this indicator does not guarantee that stock will not be pulled from the virtual warehouse; it is only used to ensure that the virtual warehouse is affected last.	PROTECTED_ IND	VARCHAR2(1)
FORECAST_ WH_IND	Alpha-numeric	1	N	Indicates whether this warehouse is forecastable. Value values are Y and N. Only warehouses with a value of Y will be visible to the forecasting tool (RDF).	FORECAST_ WH_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
REPL_IND	Alpha-numeric	1	N	Indicates whether this warehouse can be used to replenish other locations. Valid values are Y and N. Y indicates that inventory from this warehouse can be used to replenish other locations.	REPL_IND	VARCHAR2(1)
REPL_WH_LINK	Integer	10	N	Replenishable warehouse that is linked to this virtual warehouse. This link implies that the virtual warehouse is included in the net inventory calculations for the replenishable warehouse.	REPL_WH_LINK	NUMBER(10)
IB_IND	Alpha-numeric	1	N	Indicates whether the warehouse is an investment buy warehouse.	IB_IND	VARCHAR2(1)
IB_WH_LINK	Integer	10	N	Investment buy warehouse that is linked to the virtual warehouse. This link implies that the virtual warehouse is included in the net inventory calculations for the investment buy warehouse. This field should only contain a value when the IB_IND is equal to N.	IB_WH_LINK	NUMBER(10)
AUTO_IB_CLEAR	Alpha-numeric	1	N	Indicates whether the investment buy inventory should be automatically transferred to the turn (replenishable) warehouse when an order is received by the turn warehouse. Valid values are Y and N.	AUTO_IB_CLEAR	VARCHAR2(1)
FINISHER_IND	Alpha-numeric	1	N	Indicates whether the virtual warehouse performs finishing. Valid values are Y and N. Each channel must have at least one virtual warehouse that is not a finisher location (FINISHER_IND=N).	FINISHER_IND	VARCHAR2(1)
WH_NAME_SECONDARY	Alpha-numeric	150	N	Secondary description of the warehouse. This value is used to support multi-language, where the primary description may contain characters not easily sortable.	WH_NAME_SECONDARY	VARCHAR2(150)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CHANNEL_ID	Integer	4	Y	Channel to which this virtual warehouse is assigned. Within a given physical warehouse, each virtual warehouse must belong to a different channel. Valid channel IDs must exist in RMS and should be defined before warehouses are created.	CHANNEL_ID	NUMBER(4)
TSF_ENTITY_ID	Integer	10	N	Legal entity to which this virtual warehouse belongs. This field is only required when the system is operating with multiple legal entities. Valid values must exist in the RMS tsf_entity table prior to loading warehouses.	TSF_ENTITY_ID	NUMBER(10)
ORG_UNIT_ID	Numeric	15	N	Unique identifier for the Oracle Organizational ID.	ORG_UNIT_ID	NUMBER(15)
CUSTOMER_ORDER_LOC_IND	Alpha-numeric	1	N	Defines whether the virtual warehouse is customer orderable or not. Valid values:Y, N If not specified, it is defaulted to N.	CUSTOMER_ORDER_LOC_IND	VARCHAR2(1)

DC_TRANSIT_TIMES Table

File name: **DC_TRANSIT_TIMES.DAT**

Control file: **DC_TRANSIT_TIMES.CTL**

Staging table: **DC_TRANSIT_TIMES**

> **Note:** Although the RMS TRANSIT_TIMES table is loaded as part of warehouse functionality, the origin field may contain Store or Warehouse. Similarly, the destination field may contain Store or Warehouse.

Suggested post-load validation (sequence after dc_load_wh_org.ksh):

- Ensure that TRANSIT_TIMES.TRANSIT_TIMES_ID is unique.
- Ensure that TRANSIT_TIMES.DEPT is a valid DEPS.DEPT.
- Ensure that TRANSIT_TIMES.DEPT/TRANSIT_TIMES.CLASS combination exists on CLASS (if TRANSIT_TIMES.CLASS is not NULL).
- Ensure that TRANSIT_TIMES.DEPT/TRANSIT_TIMES.CLASS/ TRANSIT_TIMES.SUBCLASS combination exists on SUBCLASS (if TRANSIT_TIMES.SUBCLASS is not NULL).
- Capture the count from TRANSIT_TIMES and compare to flat file DC_TRANSIT_TIMES.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
TRANSIT_TIMES_ID	Integer	10	Y	Unique identifier of the record. This value can be sequence generated but must be unique per record loaded.	TRANSIT_TIMES_ID	NUMBER(10)
MERCH_HIER_4	Integer	4	Y	Identifier of the fourth level (from the top down) of the merchandise hierarchy (department in the base configuration) to which the transit time record applies.	DEPT	NUMBER(4)
ORIGIN	Integer	10	Y	Identifier of the supplier or location from which a shipment would originate. The identifier is a supplier ID, or a location ID with location ID type, depending on the value specified in the origin_type field.	ORIGIN	NUMBER(10)
DESTINATION	Integer	10	Y	Identifier of the location from which a shipment would be destined. The identifier is a store ID or a warehouse ID, depending on the value specified in the destination_ type field.	DESTINATION	NUMBER(10)
ORIGIN_TYPE	Alpha-numeric	2	Y	Identifier of the type of value specified in the origin field. Valid values are: ST - Stores WH - Warehouses SU - Suppliers	ORIGIN_TYPE	VARCHAR2(2)
DESTINATION_TYPE	Alpha-numeric	2	Y	Identifier of the type of value specified in the DESTINATION field. Valid values are: ST - Stores WH - Warehouses	DESTINATION_TYPE	VARCHAR2(2)
TRANSIT_TIME	Integer	4	Y	Number of days it takes for a shipment from the origin location or supplier to arrive at the destination location. This value must be expressed in terms of a whole number of days.	TRANSIT_TIME	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
MERCH_HIER_5	Integer	4	N	Identifier of the fifth level (from the top down) of the merchandise hierarchy (class in the base configuration) to which the transit time record applies. Specifying a value in this field is optional, except when a value is provided in the MERCH_HIER_6 field. If a value is not specified in this field, the records are applicable to all items that fall under the 4th level of the merchandise hierarchy. A value should be specified in this field when the transit days vary across items under the fifth level of the merchandise hierarchy.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	N	Identifier of the sixth level (from the top down) of the merchandise hierarchy (subclass in the base configuration) to which the transit time record applies. Specifying a value in this field is optional. If a value is not specified in this field, the records are applicable to all items that fall under the 4th (or 5th when populated) level of the merchandise hierarchy. A value should be specified in this field when the transit days vary across items under the sixth level of the merchandise hierarchy.	SUBCLASS	NUMBER(4)

Load Script

DC_WH.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PWH and DC_VWH staging table.

LOAD_WH– This function serves several purposes:

- It inserts data into the WH table by selecting all columns from the DC_VWH and DC_PWH staging tables, or both, and uses the defaults specified below for the columns that are not in the DC_PWH or DC_VWH tables, or that are NULL in the external tables.

  Both DC_VWH and DC_PWH tables are considered for loading data. Otherwise, only data from the DC_PWH table is loaded.
- It inserts data into the WH_ADD table. There are four total columns to be populated.

  It populates the WH_ADD pricing location with the warehouse ID (virtual warehouse ID when multi-channel is on) and the PRICING_LOC_CURR with the warehouse CURRENCY_CODE.
- It inserts data into the STOCK_LEDGER_INSERTS table. Otherwise, it inserts the physical warehouse number.

> **Note**: When multi-channel is not enabled, there is only one . file for DC_PWH data (DC_PWH.DAT). This function populates the WH, WH_ADD, and STOCK_LEDGER_INSERTS tables accordingly.
>
> **Note**: When multi-channel is enabled, there are two files for DC_PWH and DC_VWH data (DC_PWH.DAT and DC_VWH.DAT). Each physical warehouse (PWH) may have one or more virtual warehouses (VWH), so there can be one-to-many mappings between DC_PWH and DC_VWH tables. Data from both the DC_PWH and DC_VWH tables is used to insert physical warehouse records into the WH table first; then all related virtual warehouse records are inserted into the WH table. For inserts into the WH_ADD and STOCK_LEDGER_INSERTS tables, only virtual warehouse data is used.

The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_PWH to WH, WH_ADD, STOCK_LEDGER_INSERTS Column Defaults

Column Name (RMS Table	Default Value	Comments
REDIST_WH_IND	NA	From physical warehouse
ORG_ENTITY_TYPE	R	Values: R for regular warehouse; M for Importer location and X for exporter location. Default value (if NULL in external table) is R
CUSTOMER_ORDER_LOC_IND	N	IF null

Column Name (RMS Table)	Default Value	Comments
RESTRICTED_IND	N	N/A
PROTECTED_IND	N	N/A
BREAK_PACK_IND	Y	If NULL

Column Name (RMS Table)	Default Value	Comments
REDIST_WH_IND	Y	If NULL
FORECAST_WH_IND	Y	If NULL
REPL_IND	Y	If multichannel = Y then; override file value with N; otherwise, default to Y
IB_IND	No	N/A
STOCKHOLDING_IND		N if multi-channel; Y if not multi-channel
AUTO_IB_CLEAR	N	N/A
FINISHER_IND	N	This can only be Yes for virtual warehouses in a multi-channel environment; so always set it to N
PHYSICAL_WH	NA	WAREHOUSE_ID
ORG_ENTITY_TYPE	R	If multi-channel =Y then: override file value with N; otherwise, default to Y
STOCKHOLDING_IND	Y	N/A
REDIST_WH_IND	N	If NULL
PROTECTED_IND	N	If NULL
FORECAST_WH_IND	Y	If NULL
REPL_IND	N	IF NULL
IB_IND	N	IF NULL
AUTO_IB_CLEAR	N	IF NULL
FINISHER_ID	N	WAREHOUSE_ID
VAT_REGION	NA	From physical warehouse
CURRENCY_CODE	NA	From physical warehouse
ORG_HIER_TYPE	NA	From physical warehouse
ORG_HIER_VALUE	NA	From physical warehouse
DELIVERY_POLICY	NA	From physical warehouse
EMAIL	NA	From physical warehouse
DUNS_NUMBER	NA	From physical warehouse
DUNS_LOC	NA	From physical warehouse
INBOUND_HANDLING_DAYS	NA	From physical warehouse
BREAK_PACK_IND	NA	From physical warehouse

Required file to load: dc_pwh.dat. dc_vwh.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_WH_ADDR.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_WH_ADDR staging table.

LOAD_WH_ADDR– This function contains a PL/SQL block that selects from the DC_WH_ADDR staging table and inserts the data to the RMS ADDR table.

The table below defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_WH_ADDR to ADDR Column Defaults

Column Name (RMS Table)	Default Value	Comments
ADDR_KEY	System-generated	Use ADDR sequence.
MODULE	WH	N/A
SEQ_NO	1	N/A
PUBLISH_IND	N	N/A

Required file to load: dc_wh_addr.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_TRANSIT_TIMES.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TRANSIT_TIMES staging table.

LOAD_TRANSIT_TIMES– This function contains a PL/SQL block that selects from the DC_TRANSIT_TIMES staging table and inserts the data to the RMS TRANSIT_TIMES table.

Required file to load: dc_transit_times.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INS_COST_ZONE_LOCS.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_COST_ZONE_LOC– This function inserts data into the COST_ZONE and COST_ZONE_GROUP_LOC tables for the L cost level ZONE_GROUP_ID, by selecting all columns from the DC_PWH staging table. First it retrieves the ZONE_GROUP_ID for the L cost_level from the COST_ZONE_GROUP table; then it uses this ZONE_GROUP_ID to insert records for all the physical warehouses in the DC_PWH staging table into the COST_ZONE and COST_ZONE_GROUP_LOC tables.

The columns in these tables map to the DC_PWH table as follows:

- zone_ID = wh
- location = wh
- description = wh_name
- loc_type = W
- base_cost_ind = N

The same insert is performed in the COST_ZONE_GROUP_LOC table for virtual warehouses. In this insert, the values are retrieved from the DC_VWH table, and the zone_id is set to the physical_wh column value.

Required file to load: dc_pwh.dat, dc_vwh.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PROCESS_WH_ADD.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

PROCESS_WH_ADD– This function will execute function CORESVC_WH_ADD_SQL.ADD_WH to add warehouse information to WH table.

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_wh.ksh
```

> **Note:** the use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Store Overview

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- REGION
- DISTRICT
- STORE_ADD
- ADDR
- WF_CUSTOMER
- WF_CUSTOMER_GROUP

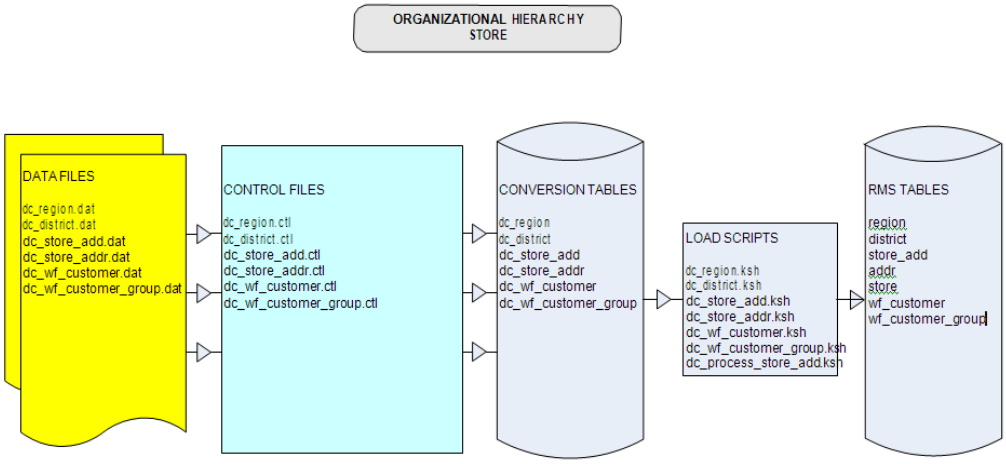
The following programs are included in this functional area:

- Load Scripts:
  - dc_district.ksh
  - dc_region.ksh
  - dc_store_add.ksh
  - dc_store_addr.ksh
  - dc_wf_customer_group.ksh
  - dc_wf_customer.ksh

- Control Files:
  - dc_district.ctl
  - dc_region.ctl
  - dc_store_add.ctl
  - dc_store_addr.ctl
  - dc_wf_customer_group.ctl
  - dc_wf_customer.ctl

Data Flow

The following diagram shows the data flow for the Organizational Hierarchy Store functional area:



Data Flow for the Organizational Hierarchy Store Functional Area

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed. File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_REGION Table

File name: **DC_REGION.DAT**

Control file: **DC_REGION.CTL**

Staging table: **DC_REGION**

Suggested post-loading validation:

- Ensure that REGION.REGION is unique.
- Ensure that REGION.AREA is a valid AREA.AREA.
- Ensure that REGION.CURRENCY_CODE (if not NULL) is a valid CURRENCIES.CURRENCY_CODE.
- Capture the count from REGION and compare to flat file DC_REGION.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
REGION	Integer	10	Y	Unique ID for the region.	REGION	NUMBER(10)
REGION_ NAME	Alpha-numeric	120	Y	Name of the region.	REGION_ NAME	VARCHAR2(120)
AREA	Integer	10	Y	ID of the area in which the region falls.	AREA	NUMBER(10)
MGR_NAME	Alpha-numeric	120	N	Name of the region manager.	MGR_ NAME	VARCHAR2(120)
CURRENCY_ CODE	Alpha-numeric	3	N	Currency under which the region operates. Valid values are in the RMS CURRENCIES table.	CURRENCY_CODE	VARCHAR2(3)

DC_DISTRICT Table

File name: **DC_DISTRICT.DAT**

Control file: **DC_DISTRICT.CTL**

Staging table: **DC_DISTRICT**

Suggested post-load validation (sequence after dc_load_store_org.ksh):

- Ensure that DISTRICT.DISTRICT is unique.
- Ensure that DISTRICT.REGION is a valid REGION.REGION.
- Ensure that DISTRICT.CURRENCY_CODE (if not NULL) is a valid CURRENCIES.CURRENCY_CODE.
- Capture the count from DISTRICT and compare to flat file DC_DISTRICT.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
DISTRICT	Integer	10	Y	Unique ID for the organization district.	DISTRICT	NUMBER(10)
DISTRICT_NAME	Alpha-numeric	120	Y	Name of the district.	DISTRICT_NAME	VARCHAR2(120)
REGION	Integer	10	Y	Unique ID for the region under which the district falls.	REGION	NUMBER(10)
MGR_NAME	Alpha-numeric	120	N	Name of the district manager.	MGR_NAME	VARCHAR2(120)
CURRENCY_CODE	Alpha-numeric	3	N	Currency under which the district operates. Valid values are in the RMS CURRENCIES table.	CURRENCY_CODE	VARCHAR2(3)

DC_STORE_ADDR Table

File name: **DC_STORE_ADDR.DAT**

Control file: **DC_STORE_ADDR.CTL**

Staging table: **DC_STORE_ADDR**

Suggested post-loading validation:

- Ensure that ADDR.KEY_VALUE_1 is a valid STORE_ADD.STORE.
- Ensure that ADDR.STATE is a valid STATE.STATE.
- Ensure that ADDR.COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.
- Capture the count from ADDR where ADDR.MODULE = ST and compare to flat file DC_STORE_ADDR.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
STORE_ID	Alpha-numeric	20	Y	Store ID for the address.	KEY_VALUE_1	VARCHAR2(20)
ADDR_TYPE	Alpha-numeric	2	Y	Type of address for this store. Valid values are: 01 - Business 02 - Postal 03 - Returns 04 - Order 05 - Invoice 06 - Remittance Additional address types can be defined in the RMS ADD_TYPE table. The required address types	ADDR_TYPE	VARCHAR2(2)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
				for a store are definable in the RMS ADD_TYPE_MODULE table, where MODULE = ST for company stores and WFST for franchise stores.		
PRIMARY_ADDR_IND	Alpha-numeric	1	Y	Indicates whether this is the primary address for the address type.	PRIMARY_ADDR_IND	VARCHAR2(1)
CONTACT_NAME	Alpha-numeric	120	N	Name of the contact at this address.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	N	Phone number of the contact at the address.	CONTACT_PHONE	VARCHAR2(20)
CONTACT_FAX	Alpha-numeric	20	N	Fax number of the contact at the address.	CONTACT_FAX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	E-mail of the contact at the address.	CONTACT_EMAIL	VARCHAR2(100)
CONTACT_TELEX	Alpha-numeric	20	N	Telex number of the contact at the address.	CONTACT_TELEX	VARCHAR2(20)
ADDR_LINE_1	Alpha-numeric	240	Y	First line of the address.	ADD_1	VARCHAR2(240)
ADDR_LINE_2	Alpha-numeric	240	N	Second line of the address.	ADD_2	VARCHAR2(240)
ADDR_LINE_3	Alpha-numeric	240	N	Third line of the address.	ADD_3	VARCHAR2(240)
CITY	Alpha-numeric	120	Y	City of the address.	CITY	VARCHAR2(120)
COUNTY	Alpha-numeric	250	N	County in which the city is located.	COUNTY	VARCHAR2(250)
STATE	Alpha-numeric	3	N	State in which the city is located.	STATE	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	ZIP code of the address.	POST	VARCHAR2(30)
COUNTRY_ID	Alpha-numeric	3	Y	Country ID. Valid values are in the RMS COUNTRY table.	COUNTRY_ID	VARCHAR2(3)
JURISDICTION_CODE	Alpha-numeric	10	N	Jurisdiction code for the address	JURISDICTION_CODE	VARCHAR2(10)

DC_STORE_ADD Table

File name: **DC_STORE_ADD.DAT**

Control file: **DC_STORE_ADD.CTL**

Staging table: **DC_STORE_ADD**

Suggested post-loading validation:

- Ensure that STORE_ADD.STORE is unique and does not exist on STORE.
- Ensure that STORE_ADD.TSFZONE (if not NULL) is a valid TSFZONE.TRANSFER_ZONE.
- Ensure that STORE_ADD.CURRENCY_CODE is a valid CURRENCIES.CURRENCY_CODE.
- Ensure that STORE_ADD.LANG is a valid LANG.LANG.
- Ensure that STORE_ADD.DISTRICT is a valid DISTRICT.DISTRICT.
- Ensure that STORE_ADD.DEFAULT_WH (if not NULL) is a valid WH.WH, where WH.STOCKHOLDING_IND = Y.
- Ensure that STORE_ADD.ORG_UNIT_ID (if not NULL) is a valid ORG_UNIT.ORG_UNIT_ID.
- Ensure that STORE_ADD.STORE_FORMAT (if not NULL) is a valid STORE_FORMAT.STORE_FORMAT.

FILE FORMAT					**STAGING TABLE DEFINITION**	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
STORE	Integer	10	Y	Unique ID of the store.	STORE	NUMBER(10)
STORE_NAME	Alpha-numeric	150	Y	Name of the store.	STORE_NAME	VARCHAR2(150)
STORE_NAME10	Alpha-numeric	10	Y	10-character abbreviation of the store name.	STORE_NAME10	VARCHAR2(10)
STORE_NAME3	Alpha-numeric	3	Y	3-character abbreviation of the store name.	STORE_NAME3	VARCHAR2(3)
STORE_CLASS	Alpha-numeric	1	Y	Code letter indicating the class of which the store is a member: A, B, C, D, or E.	STORE_CLASS	VARCHAR2(1)
STORE_MGR_NAME	Alpha-numeric	120	Y	Name of the store manager.	STORE_MGR_NAME	VARCHAR2(120)
STORE_OPEN_DATE	Alpha-numeric	9	Y	Date the store opened. Date format is DDMONYYYY (for example, 02JAN2011).	STORE_OPEN_DATE	DATE
STOCKHOLDING_IND	Alpha-numeric	1	N	Indicates whether the store can hold stock, default N.	STOCKHOLDING_IND	VARCHAR2(1)
DISTRICT	Integer	10	Y	ID of the district in which the store is located. Must be a valid district ID.	DISTRICT	NUMBER(10)
START_ORDER_DAYS	Integer	3	Y	Days before the store open date that the store can start accepting orders.	START_ORDER_DAYS	NUMBER(3)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CURRENCY_CODE	Alpha-numeric	3	Y	Currency in which the store operates. Must be a valid currency code in the RMS CURRENCIES table.	CURRENCY_CODE	VARCHAR2(3)
LANG	Integer	6	Y	Language in which the store operates. Must be a valid language code that exists in the RMS LANG table.	LANG	NUMBER(6)
COPY_REPL_IND	Alpha-numeric	1	N	Indicates whether the replenishment information set up for the like store will be copied (Y or N, default N).	COPY_REPL_IND	VARCHAR2(1)
TRAN_NO_GENERATED	Alpha-numeric	6	Y	Indicates whether the transaction ID is unique by store or by store/register (S or R).	TRAN_NO_GENERATED	VARCHAR2(6)
INTEGRATED_POS_IND	Alpha-numeric	1	Y	Indicates whether the POS at the store is integrated(Y or N).	INTEGRATED_POS_IND	VARCHAR2(1)
COPY_ACTIVITY_IND	Alpha-numeric	1	N	Indicates whether the like store's closing date schedule should be copied in the creation of a new store based on a like store (Y or N, default N).	COPY_ACTIVITY_IND	VARCHAR2(1)
COPY_DLVRY_IND	Alpha-numeric	1	N	Indicates whether the like store's delivery schedule should be copied in the creation of a new store based on a like store (Y or N, default N).	COPY_DLVRY_IND	VARCHAR2(1)
STORE_NAME_SECONDARY	Alpha-numeric	150	N	Secondary name of the store. This field can be populated only when SYSTEM_OPTIONS. SECONDARY_DESC_IND = Y.	STORE_NAME_SECONDARY	VARCHAR2(150)
STORE_CLOSE_DATE	Alpha-numeric	9	N	Date the store closed. Date format is DDMONYYYY (for example, 02JAN2011).	STORE_CLOSE_DATE	DATE(7)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
ACQUIRED_DATE	Alpha-numeric	9	N	Date the store was acquired. Date format is DDMONYYYY.	ACQUIRED_DATE	DATE(7)
REMODEL_DATE	Alpha-numeric	9	N	Last date the store was remodeled. Date format is DDMONYYYY.	REMODEL_DATE	DATE(7)
FAX_NUMBER	Alpha-numeric	20	N	Fax number of the contact at the store.	FAX_NUMBER	VARCHAR2(20)
PHONE_NUMBER	Alpha-numeric	20	N	Phone number of the store.	PHONE_NUMBER	VARCHAR2(20)
EMAIL	Alpha-numeric	100	N	E-mail of the contact at the store.	EMAIL	VARCHAR2(100)
TOTAL_SQUARE_FT	Integer	8	N	Total square feet of the store.	TOTAL_SQUARE_FT	NUMBER(8)
SELLING_SQUARE_FT	Integer	8	N	Square feet dedicated to selling merchandise.	SELLING_SQUARE_FT	NUMBER(8)
LINEAR_DISTANCE	Integer	8	N	Total merchandise area of the store.	LINEAR_DISTANCE	NUMBER(8)
VAT_REGION	Integer	4	N	Number of the value added tax region in which this store is located. Required if VAT_INCLUDE_IND = N. Valid values are found in the RMS VAT_REGION table.	VAT_REGION	NUMBER(4)
VAT_INCLUDE_IND	Alpha-numeric	1	N	Indicates whether value added tax is included in the retail prices for the store. Valid values are Y or N, default N.	VAT_INCLUDE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CHANNEL_ID	Integer	4	N	In a multi-channel environment, this contains the channel with which the store is associated. Valid values can be found in the CHANNELS table. This is required for a multi-channel environment; the value must be a valid channel ID.	CHANNEL_ID	NUMBER(4)
STORE_FORMAT	Integer	4	N	Number indicating the format of the store. Valid values are found in the store format table.	STORE_FORMAT	NUMBER(4)
MALL_NAME	Alpha-numeric	120	N	Name of the mall in which the store is located.	MALL_NAME	VARCHAR2(120)
TRANSFER_ZONE	Integer	4	N	Transfer zone in which the store is located. Valid values are located in the TSFZONE table.	TRANSFER_ZONE	NUMBER(4)
DEFAULT_WH	Integer	10	N	Number of the warehouse that can be used as the default for creating cross-dock masks. This determines which stores are associated with or sourced from a warehouse. It holds only virtual warehouses in a multi-channel environment. Otherwise, this field is NULL.	DEFAULT_WH	NUMBER(10)
STOP_ORDER_DAYS	Integer	3	N	Number of days before a store closing that the store will stop accepting orders. This column is used when the STORE_CLOSE_DATE is defined.	STOP_ORDER_DAYS	NUMBER(3)
DUNS_NUMBER	Alpha-numeric	9	N	Dun and Bradstreet number to identify the store.	DUNS_NUMBER	VARCHAR2(9)
DUNS_LOC	Alpha-numeric	4	N	Dun and Bradstreet number to identify the location.	DUNS_LOC	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SISTER_STORE	Integer	10	N	Store number used to relate the current store to the historical data of an existing store.	SISTER_STORE	NUMBER(10)
TSF_ENTITY_ID	Integer	10	N	Legal entity in which the store is located. This field is required in a multi-channel environment. Foreign key to the TSF_ENTITY table	TSF_ENTITY_ID	NUMBER(10)
ORG_UNIT_ID	Numeric	15	N	Unique identifier for the Oracle Organizational ID.	ORG_UNIT_ID	NUMBER(15)
STORE_TYPE	Alpha-numeric	6	N	Type of store. Valid values are: C – Company F - Franchise	STORE_TYPE	VARCHAR2(6)
WF_CUSTOMER_ID	Integer	10	N	Unique ID of the franchise store. This column will be null if the store_type = C.	WF_CUSTOMER_ID	NUMBER(10)
AUTO_APPROVE_ORDERS_IND	Alpha-numeric	1	N	Indicates whether electronic orders to this store should be approved automatically if the order can be fulfilled. Valid values are Y and N.	AUTO_APPROVE_ORDERS_IND	VARCHAR2(1)
TIMEZONE_NAME	Alpha-numeric	64	Y	Name of the Time Zone in which the store is located.	TIMEZONE_NAME	VARCHAR2(64)
CUSTOMER_ORDER_LOC	Alpha-numeric	1	N	Defines whether the store is customer orderable or not. Valid values: Company Stores : Y, N Franchise Stores : Y, N	CUSTOMER_ORDER_LOC_IND	VARCHAR2(1)

DC_WF_CUSTOMER Table

File name: **DC_WF_CUSTOMER.DAT**

Control file: **DC_WF_CUSTOMER.CTL**

Staging table: **DC_WF_CUSTOMER**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WF_CUSTOMER_ID	Integer	10	Y	Unique ID of the franchise store.	WF_CUSTOMER_ID	NUMBER(10)
WF_CUSTOMER_NAME	Alpha-numeric	120	Y	Name of the franchise store.	WF_CUSTOMER_NAME	VARCHAR2(120)
CREDIT_IND	Alpha-numeric	1	Y	Indicates whether the franchise store has good credit standing. Valid values are Y and N.	CREDIT_IND	VARCHAR2(1)
WF_CUSTOMER_GROUP_ID	Integer	10	Y	Unique ID of a customer group associated with the franchise store.	WF_CUSTOMER_GROUP_ID	NUMBER(10)

DC_WF_CUSTOMER_GROUP Table

File name: **DC_WF_CUSTOMER_GROUP.DAT**

Control file: **DC_WF_CUSTOMER_GROUP.CTL**

Staging table: **DC_WF_CUSTOMER_GROUP**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WF_CUSTOMER_GROUP_ID	Integer	10	Y	Unique ID of the customer group.	WF_CUSTOMER_GROUP_ID	NUMBER(10)
WF_CUSTOMER_GROUP_NAME	Alpha-numeric	120	Y	Name of the customer group.	WF_CUSTOMER_GROUP_NAME	VARCHAR2(120)

Load Scripts

DC_REGION.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_REGION staging table.

LOAD_REGION– This function contains a PL/SQL block that selects from the DC_REGION staging table and inserts the data to the RMS REGION table.

Required file to load: dc_region.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_DISTRICT.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_DISTRICT staging table.

LOAD_DISTRICT– This function contains a PL/SQL block that selects from the DC_DISTRICT staging table and inserts the data to the RMS DISTRICT table.

Required file to load: dc_district.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_STORE_ADDR.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_STORE_ADDR staging table.

LOAD_STORE_ADDRESS– This function contains a PL/SQL block that selects from the DC_STORE_ADDR staging table and inserts the data to the RMS ADDR table. The table below defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_STORE_ADDR to ADDR Column Defaults

Column Name (RMS Table)	Default Value	Comments
ADDR_KEY	System-generated	NA
MODULE	ST	NA
SEQ_NO	1	NA
PUBLISH_IND	N	NA

Required file to load: dc_store_addr.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_STORE_ADD.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_STORE_ADD staging table.

LOAD_STORE_ADD– This function contains a PL/SQL block that selects from the DC_STORE_ADD staging table and inserts the data to the RMS STORE_ADD table. The table below defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_STORE_ADD to STORE_ADD Column Defaults

Column Name (RMS Table)	Default Value	Comments
STOCKHOLDING_IND	Y	Y or N
COPY_REPL_IND	N	Y or N
COPY_ACTIVITY_IND	N	Y or N
COPY_DLVRY_IND	N	Y or N
VAT_INCLUDE_IND	N	Y or N
CUSTOMER_ORDER_LOC_IND	N	Defaults to N and NULL for company stores and stockholding franchise stores.

Required file to load: dc_store_add.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_WF_CUSTOMER.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_WF_CUSTOMER staging table.

LOAD_ORG_UNIT– This function selects all columns from the DC_WF_CUSTOMER staging table and inserts data into the WF_CUSTOMER table. The DC_WF_CUSTOMER staging table maps exactly to the RMS WF_CUSTOMER table.

Required file to load: dc_wf_customer.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PROCESS_STORE_ADD.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

PROCESS_STORE_ADD– This function will execute function CORESVC_STORE_ADD_SQL.ADD_STORE to add store information to STORE table.

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Post-Loading Requirements

After using the data conversion toolset for this functional area, there are additional tables that must be loaded manually before you proceed with data conversion for subsequent functional areas, because of data dependencies.

Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following are **required** tables that require manual data loading:

- DEPT_CHRG_HEAD
- DEPT_CHRG_DETAIL
- STORE_HIERARCHY
- COST_ZONE_GROUP (zone level pricing)

> **Note**: Location level COST_ZONE_GROUP should have been created by the seed data installation. See "Appendix: Seed Data Installation" for more information.

- COST_ZONE
- COST_ZONE_GROUP_LOC
- RPM requirements:
  - RPM_ZONE_GROUP_TYPE
  - RPM_ZONE_GROUP
  - RPM_ZONE
  - RPM_ZONE_LOCATION

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_region.ksh
```

Note The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

6

Suppliers

This chapter describes data conversion for the following RMS tables, listed in the order that they must be loaded:

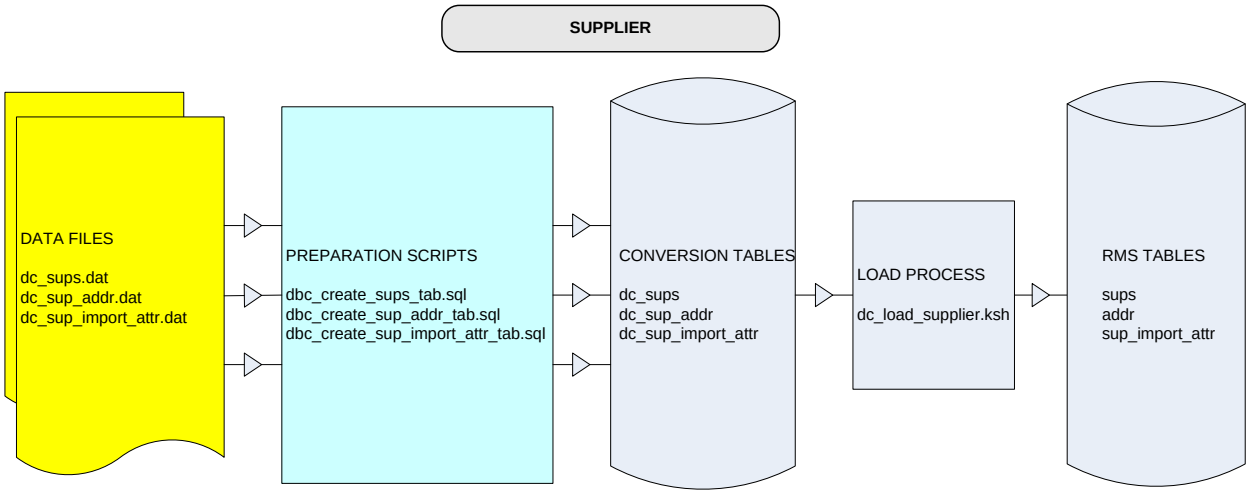
- SUPS
- ADDR (for supplier addresses)
- SUP_IMPORT_ATTR
- PARTNER

The following programs are included in the Suppliers functional area:

- Load Scripts:
  - dc_sups.ksh
  - dc_sup_addr.ksh
  - dc_sup_import_attr.ksh
- Control Files:
  - dc_sups.ctl
  - dc_sup_sddr.ctl
  - dc_sup_import_attr.ctl

Data Flow

The following diagram shows the data flow for the Suppliers functional area:



Data Flow for the Suppliers Functional Area

Prerequisites

Before you begin using the data conversion toolset for Suppliers, you must complete data conversion for the following functional areas:

- Core
- Merchandise Hierarchy
- Organizational Hierarchy

There are tables that must be loaded manually, because of data dependencies for auto-loading within this functional area. Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

The following **required** tables must be loaded manually:

- PARTNER (required types: AG=agents, BK=advising or issuing banks, FA=factory)
- OUTLOC (required types: DP=discharge ports, LP=lading ports)

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

The dc_load_supplier.ksh script calls each of the SQL scripts in a specific order. The SQL scripts create external Oracle tables from flat file feeds and load data into the Oracle Retail Merchandising database.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

Suppliers--DC_SUPS Table

File name: **DC_SUPS.DAT**

Control file: **DC_SUPS.CTL**

Staging table: **DC_SUPS**

Suggested post-loading validation:

- Ensure that SUPS.SUPPLIER is unique.
- Ensure that SUPS.CURRENCY_CODE is a valid CURRENCIES.CURRENCY_CODE.
- Ensure that SUPS.TERMS is a valid TERMS_HEAD.TERMS.
- Ensure that SUPS.FREIGHT_TERMS is a valid FREIGHT_TERMS.FREIGHT_TERMS.
- Ensure that SUPS.LANG (if not NULL) is a valid LANG.LANG.
- Capture supplier number from SUPS where SUPS.BRACKET_COSTING_IND = Y to ensure that SUP_BRACKET_COST rows are added manually.
- Capture supplier number from SUPS where SUPS.RET_ALLOW_IND = Y to ensure that row for the supplier with ADDR_TYPE = 03 exists in ADDR.
- Capture the count from SUPS and compare to flat file DC_SUPS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SUPPLIER	Integer	10	Y	Unique number for the supplier.	SUPPLIER	NUMBER(10)
SUP_NAME	Alpha-numeric	240	Y	Name of the supplier.	SUP_NAME	VARCHAR2(240)
SUP_NAME_SECONDARY	Alpha-numeric	240	N	Secondary name of the supplier. This field can only be populated when SYSTEM_OPTIONS.SECONDARY_DESC_IND = Y.	SUP_NAME_SECONDARY	VARCHAR2(240)
CONTACT_NAME	Alpha-numeric	120	Y	Name of contact at the supplier.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	Y	Phone number of the contact at the supplier.	CONTACT_PHONE	VARCHAR2(20)
CONTACT_FAX	Alpha-numeric	20	N	Fax number of the contact at the supplier.	CONTACT_FAX	VARCHAR2(20)
CONTACT_PAGER	Alpha-numeric	20	N	Pager number of the contact at the supplier.	CONTACT_PAGER	VARCHAR2(20)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
QC_IND	Alpha-numeric	1	Y	Indicates whether orders from this supplier default as requiring quality control. A value of Y means that all orders from this supplier require quality control, unless overridden by the user when the order is created. An N in this field means that quality control will not be required, unless indicated by the user during order creation.	QC_IND	VARCHAR2(1)
QC_PCT	Float	12,4	N	Percentage of items per receipt that will be marked for quality checking.	QC_PCT	NUMBER(12,4)
QC_FREQ	Integer	2	N	Frequency with which items per receipt will be marked for quality checking.	QC_FREQ	NUMBER(2)
VC_IND	Alpha-numeric	1	Y	Indicates whether orders from this supplier default as requiring vendor control. A value of Y means that all orders from this supplier will require vendor control. N means that vendor control will not be required.	VC_IND	VARCHAR2(1)
VC_PCT	Float	12,4	N	Percentage of items per receipt that will be marked for vendor checking.	VC_PCT	NUMBER(12,4)
VC_FREQ	Integer	2	N	Frequency with which items per receipt will be marked for vendor checking.	VC_FREQ	NUMBER(2)
CURRENCY_CODE	Alpha-numeric	3	Y	Currency the supplier uses for business transactions. Valid values are in the RMS CURRENCIES table.	CURRENCY_CODE	VARCHAR2(3)
LANG	Integer	6	N	Supplier's preferred language. This field is provided for custom purchase orders in a specified language. Valid values are stored in the LANG table in RMS.	LANG	NUMBER(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
TERMS	Alpha-numeric	15	Y	Indicator identifying the sales terms that default when an order is created for the supplier. These terms specify when payment is due and if there are any discounts for early payment. Valid values are in the RMS TERMS_HEAD table.	TERMS	VARCHAR2(15)
FREIGHT_TERMS	Alpha-numeric	30	Y	Indicator that references the freight terms that default when an order is created for the supplier. Valid values are in the RMS FREIGHT_TERMS table.	FREIGHT_TERMS	VARCHAR2(30)
RET_ALLOW_IND	Alpha-numeric	1	Y	Indicates whether the supplier accepts returns. Valid values are Y and N.	RET_ALLOW_IND	VARCHAR2(1)
RET_AUTH_REQ	Alpha-numeric	1	Y	Indicates whether returns must be accompanied by an authorization number when sent back to the vendor. Valid values are Y and N.	RET_AUTH_REQ	VARCHAR2(1)
RET_MIN_DOL_AMT	Numeric	20,4	N	Contains a value if the supplier requires a minimum dollar amount to be returned in order to accept the return. Returns of less than this amount will not be processed by the system. This field is stored in the supplier's currency.	RET_MIN_DOL_AMT	NUMBER(20,4)
RET_COURIER	Alpha-numeric	250	N	Name of the courier that should be used for all returns to the supplier.	RET_COURIER	VARCHAR2(250)
DEFAULT_HANDLING_PCT	Numeric	12,4	N	Percentage multiplied by the total order cost to determine the handling cost for the return.	HANDLING_PCT	NUMBER(12,4)
EDI_PO_IND	Alpha-numeric	1	Y	Indicates whether purchase orders will be sent to the supplier through Electronic Data Interchange. Valid values are Y and N.	EDI_PO_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
EDI_PO_CHG	Alpha-numeric	1	Y	Indicates whether purchase order changes will be sent to the supplier through Electronic Data Interchange. Valid values are Y and N.	EDI_PO_CHG	VARCHAR2(1)
EDI_PO_CONFIRM	Alpha-numeric	1	Y	Indicates whether this supplier will send acknowledgment of a purchase orders sent through Electronic Data Interchange. Valid values are Y and N.	EDI_PO_CONFIRM	VARCHAR2(1)
EDI_ASN	Alpha-numeric	1	Y	Indicates whether this supplier will send Advance Shipment Notifications electronically. Valid values are Y and N.	EDI_ASN	VARCHAR2(1)
EDI_SALES_RPT_FREQ	Alpha-numeric	1	N	EDI sales report frequency for this supplier. Valid values are: D - Sales and stock information will be downloaded daily. W - Sales and stock information will be downloaded weekly.	EDI_SALES_RPT_FREQ	VARCHAR2(1)
EDI_SUPP_AVAILABLE_IND	Alpha-numeric	1	Y	Indicates whether the supplier will send availability through EDI.	EDI_SUPP_AVAILABLE_IND	VARCHAR2(1)
EDI_CONTRACT_IND	Alpha-numeric	1	Y	Indicates whether contracts will be sent to the supplier through EDI.	EDI_CONTRACT_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
EDI_CHANNEL_ID	Integer	4	N	Used only in a multi-channel environment. If the supplier is an EDI supplier and supports vendor-initiated ordering, this field contains the channel ID of the channel to which all inventory for these types of orders will flow. This field is used when a vendor-initiated order is created for a physical warehouse, to determine the virtual warehouse within the physical warehouse to which the inventory will flow. The virtual warehouse belonging to the indicated channel will be used.	EDI_CHANNEL_ID	NUMBER(4)
REPLEN_APPROVAL_IND	Alpha-numeric	1	Y	Indicates whether contract orders for the supplier should be created in approved status. Valid values are Y and N.	REPLEN_APPROVAL_IND	VARCHAR2(1)
SHIP_METHOD	Alpha-numeric	6	N	Unique number for the supplier.	SHIP_METHOD	VARCHAR2(6)
PAYMENT_METHOD	Alpha-numeric	6	N	Name of the supplier.	PAYMENT_METHOD	VARCHAR2(6)
CONTACT_TELEX	Alpha-numeric	20	N	Secondary name of the supplier. This field can only be populated when SYSTEM_OPTIONS.SECONDARY_DESC_IND = Y.	CONTACT_TELEX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	Name of contact at the supplier.	CONTACT_EMAIL	VARCHAR2(100)
SETTLEMENT_CODE	Alpha-numeric	1	Y	Phone number of the contact at the supplier.	SETTLEMENTCODE	VARCHAR2(1)
PRE_MARK_IND	Alpha-numeric	1	Y	Fax number of the contact at the supplier.	PRE_MARK_IND	VARCHAR2(1)
AUTO_APPR_INVC_IND	Alpha-numeric	1	Y	Pager number of the contact at the supplier.	AUTO_APPR_INVC_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
FREIGHT_CHARGE_IND	Alpha-numeric	1	Y	Indicates whether orders from this supplier default as requiring quality control. A value of Y means that all orders from this supplier require quality control, unless overridden by the user when the order is created. An N in this field means that quality control will not be required, unless indicated by the user during order creation.	FREIGHT_CHARGE_IND	VARCHAR2(1)
BACKORDER_IND	Alpha-numeric	1	Y	Percentage of items per receipt that will be marked for quality checking.	BACKORDER_IND	VARCHAR2(1)
VAT_REGION	Integer	4	N	Frequency with which items per receipt will be marked for quality checking.	VAT_REGION	NUMBER(4)
INV_MGMT_LVL	Alpha-numeric	6	N	Indicates whether orders from this supplier default as requiring vendor control. A value of Y means that all orders from this supplier will require vendor control. N means that vendor control will not be required.	INV_MGMT_ LVL	VARCHAR2(6)
SERVICE_PERF_REQ_IND	Alpha-numeric	1	Y	Percentage of items per receipt that will be marked for vendor checking.	SERVICE_PERF_REQ_IND	VARCHAR2(1)
DELIVERY_POLICY	Alpha-numeric	6	Y	Frequency with which items per receipt will be marked for vendor checking.	DELIVERY_POLICY	VARCHAR2(6)
COMMENT_DESC	Alpha-numeric	2000	N	Currency the supplier uses for business transactions. Valid values are in the RMS CURRENCIES table.	COMMENT_DESC	VARCHAR2(2000)
DEFAULT_ITEM_LEAD_TIME	Integer	4	N	Supplier's preferred language. This field is provided for custom purchase orders in a specified language. Valid values are stored in the LANG table in RMS.	DEFAULT_ITEM_LEAD_TIME	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
DUNS_NUMBER	Alpha-numeric	9	N	Indicator identifying the sales terms that default when an order is created for the supplier. These terms specify when payment is due and if there are any discounts for early payment. Valid values are in the RMS TERMS_HEAD table.	DUNS_NUMBER	VARCHAR2(9)
DUNS_LOC	Alpha-numeric	4	N	Indicator that references the freight terms that default when an order is created for the supplier. Valid values are in the RMS FREIGHT_TERMS table.	DUNS_LOC	VARCHAR2(4)
DEFAULT_VMI_ORDER_STATUS	Alpha-numeric	6	N	Indicates whether returns must be accompanied by an authorization number when sent back to the vendor. Valid values are Y and N.	VMI_ORDER_STATUS	VARCHAR2(6)
DSD_IND	Alpha-numeric	1	Y	Contains a value if the supplier requires a minimum dollar amount to be returned in order to accept the return. Returns of less than this amount will not be processed by the system. This field is stored in the supplier's currency.	DSD_IND	VARCHAR2(1)
SUPPLIER_PARENT	Numeric	10	N	This has a value of Supplier number for the Supplier Site. For Suppliers, this field will be NULL.	SUPPLIER_PARENT	NUMBER(10)
SUP_QTY_LEVEL	Alpha-numeric	6		This field is not nullable. Valid values are CA (cases) and EA (eaches). Default = EA	SUP_QTY_LEVEL	VARCHAR2(6)

Supplier Address - DC_SUP_ADDR Table

File name: **DC_SUP_ADDR.DAT**

Control file: **DC_SUP_ADDR.CTL**

Staging table: **DC_SUP_ADDR**

Suggested post-loading validation:

- Ensure that ADDR.KEY_VALUE_1 is a valid SUPS.SUPPLIER.
- Ensure that ADDR.STATE is a valid STATE.STATE.
- Ensure that ADDR.COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.
- Ensure that every SUPS.SUPPLIER with SUPS.RET_ALLOW_IND = Y has a row in ADDR with ADDR.MODULE = SUPP and ADDR.ADDR_TYPE = 03.
- Ensure that every SUPS.SUPPLIER has a row in ADDR with ADDR.MODULE = SUPP, and ADDR.ADDR_TYPE in the set of all ADD_TYPE_MODULE.ADDRESS_TYPE, with ADD_TYPE_MODULE.MODULE = SUPP and ADD_TYPE_MODULE.MANDATORY_IND = Y.
- Ensure every ADDR.ADDR_TYPE where ADDR.MODULE = SUPP is a valid ADD_TYPE_MODULE.ADDRESS_TYPE with ADD_TYPE_MODULE.MODULE = SUPP.
- Capture the count from ADDR where ADDR.MODULE = SUPP and compare to flat file DC_SUP_ADDR.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SUPPLIER_ID	Integer	10	Y	Unique ID of the supplier.	KEY_VALUE_1	NUMBER(10)
ADDR_TYPE	Alpha-numeric	2	Y	Type of address for this supplier. Valid values are: 01 - Business 02 - Postal 03 - Returns 04 - Order 05 - Invoice 06 - Remittance Additional address types can be defined in the RMS ADD_TYPE table. The required address types for a supplier are definable in the RMS ADD_TYPE_MODULE table where MODULE = SUPP.	ADDR_TYPE	VARCHAR2(2)
PRIMARY_ADDR_IND	Alpha-numeric	1	Y	Indicates whether the address is the primary address for this address type.	PRIMARY_ADDR_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CONTACT_NAME	Alpha-numeric	120	Y	Name of the contact at this address.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	N	Phone number of the contact at this address.	CONTACT_PHONE	VARCHAR2(20)
CONTACT_TELEX	Alpha-numeric	20	N	Telex number of the contact at this address.	CONTACT_TELEX	VARCHAR2(20)
CONTACT_FAX	Alpha-numeric	20	N	Fax number of the contact at this address.	CONTACT_FAX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	E-mail of the contact at this address.	CONTACT_EMAIL	VARCHAR2(100)
ADDR_LINE_1	Alpha-numeric	240	Y	First line of the address.	ADD_1	VARCHAR2(240)
ADDR_LINE _2	Alpha-numeric	240	N	Second line of the address.	ADD_2	VARCHAR2(240)
ADDR_LINE _3	Alpha-numeric	240	N	Third line of the address.	ADD_3	VARCHAR2(240)
CITY	Alpha-numeric	120	Y	Name of the city of this address.	CITY	VARCHAR2(120)
COUNTY	Alpha-numeric	250	N	County of the address.	COUNTY	VARCHAR2(250)
STATE	Alpha-numeric	3	N	State abbreviation of the address.	STATE	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	ZIP code.	POST	VARCHAR2(30)
COUNTRY_ID	Alpha-numeric	3	Y	Country code. Valid values are in the COUNTRY table.	COUNTRY_ID	VARCHAR2(3)
JURISDICTION_CODE	Alpha-numeric	10	N	Jurisdiction code for the address	JURISDICTION_CODE	VARCHAR2(10)

Supplier Import Attributes - DC_SUP_IMPORT_ATTR Table

File name: **DC_SUP_IMPORT_ATTR.DAT**

Control file: **DC_SUP_IMPORT_ATTR.CTL**

Staging table: **DC_SUP_IMPORT_ATTR**

Suggested post-loading validation:

- Ensure that SUP_IMPORT.ATTR.AGENT is a valid PARTNER.PARTNER_ID with PARTNER_TYPE = AG.
- Ensure that SUP_IMPORT.ATTR.ADVISING_BANK is a valid PARTNER.PARTNER_ID with PARTNER_TYPE = BK.

- Ensure that SUP_IMPORT.ATTR.ISSUING_BANK is a valid PARTNER.PARTNER_ID with PARTNER_TYPE = BK.
- Ensure that SUP_IMPORT.ATTR.LADING_PORT is a valid OUTLOC.OUTLOC_ID with OUTLOC.OUTLOC_TYPE = LP.
- Ensure that SUP_IMPORT.ATTR.DISCHARGE_PORT is a valid OUTLOC.OUTLOC_ID with OUTLOC.OUTLOC_TYPE = DP.
- Ensure that SUP_IMPORT_ATTR.PLACE_OF_EXPIRY is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = LCPE.
- Ensure that SUP_IMPORT_ATTR.DRAFTS_AT is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = LCDA.
- Ensure that SUP_IMPORT_ATTR.PRESENTATION_TERMS is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = LCPT.
- Ensure that SUP_IMPORT_ATTR.PARTNER_1 is a valid PARTNER.PARTNER_ID with the same partner type as SUP_IMPORT_ATTR.PARTNER_TYPE_1.
- Ensure that SUP_IMPORT_ATTR.PARTNER_2 is a valid PARTNER.PARTNER_ID with the same partner type as SUP_IMPORT_ATTR.PARTNER_TYPE_2.
- Ensure that SUP_IMPORT_ATTR.PARTNER_3 is a valid PARTNER.PARTNER_ID with the same partner type as SUP_IMPORT_ATTR.PARTNER_TYPE_3.
- Capture the count from SUP_IMPORT_ATTR and compare to flat file DC_SUP_IMPORT_ATTR.DAT to ensure that all rows are loaded.

FILE FORMAT					**STAGING TABLE DEFINITION**	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SUPPLIER	Integer	10	Y	Unique ID of the supplier.	SUPPLIER	NUMBER(10)
AGENT	Alpha-numeric	10	N	Agent associated with the supplier.	AGENT	VARCHAR2(10)
ADVISING_BANK	Alpha-numeric	10	N	Bank advising the Letter of Credit.	ADVISING_BANK	VARCHAR2(10)
ISSUING_BANK	Alpha-numeric	10	N	Bank issuing the letter of credit.	ISSUING_BANK	VARCHAR2(10)
LADING_PORT	Alpha-numeric	5	N	Identification number of the supplier's Lading Port.	LADING_PORT	VARCHAR2(5)
DISCHARGE_PORT	Alpha-numeric	5	N	Identification number of the supplier's discharge port.	DISCHARGE_PORT	VARCHAR2(5)
MFG_ID	Alpha-numeric	18	N	Manufacturer's tax identification number.	MFG_ID	VARCHAR2(18)
RELATED_IND	Alpha-numeric	1	Y	Indicates whether the supplier is related to the company. Valid values are Y and N.	RELATED_IND	VARCHAR2(1)
BENEFICIARY_IND	Alpha-numeric	1	Y	Indicates whether this supplier can be a beneficiary. Valid values are Y and N.	BENEFICIARY_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WITH_RECOURSE_IND	Alpha-numeric	1	Y	Conditional payment on the part of the bank, as instructed by the buyer. Valid values are Y and N.	WITH_RECOURSE_IND	VARCHAR2(1)
REVOCABLE_IND	Alpha-numeric	1	Y	Indicates whether the letter of credit is revocable. If this is Y, the letter of credit can be amended or cancelled at any time by the buyer or buyer's bank. If this is 'N', the letter of credit has to have both buyer and seller approval to do anything.	REVOCABLE_IND	VARCHAR2(1)
VARIANCE_PCT	Numeric	12,4	N	Allowed currency variance percentage for the letter of credit. For example, if the variance percent is 5, the letter of credit can be underpaid or overpaid by 5 percent.	VARIANCE_PCT	NUMBER(12,4)
LC_NEG_DAYS	Integer	3	N	Number of days to negotiate documents.	LC_NEG_DAYS	NUMBER(3)
PLACE_OF_EXPIRY	Alpha-numeric	6	N	Place where the letter of credit will expire. Valid values are: 01 - Issuing Bank 02 - Advising Bank 03 - Miami 04 - New York 05 – Los Angeles	PLACE_OF_EXPIRY	VARCHAR2(6)
DRAFTS_AT	Alpha-numeric	6	N	Terms of draft (or when payment is to be made) for the letter of credit. Valid values are: 01 - At sight 02 - 30 Days 03 - 60 Days	DRAFTS_AT	VARCHAR2(6)
PRESENTATION_TERMS	Alpha-numeric	6	N	Terms of presentation (for example, “to the order of any bank” or “to XYZ Bank”). Valid values are: P - By payment A - By acceptance N - By negotiation	PRESENTATION_TERMS	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
FACTORY	Alpha-numeric	10	N	Factory partner ID for the factory partner type.	FACTORY	VARCHAR2(10)
PARTNER_TYPE_1	Alpha-numeric	6	N	Partner type of the first additional partner. Valid values are in the RMS PARTNER table.	PARTNER_TYPE_1	VARCHAR2(6)
PARTNER_1	Alpha-numeric	10	N	Partner ID of the first additional partner. Valid values are in the RMS PARTNER table.	PARTNER_1	VARCHAR2(10)
PARTNER_TYPE_2	Alpha-numeric	6	N	Partner type of the second additional partner. Valid values are in the RMS PARTNER table.	PARTNER_TYPE_2	VARCHAR2(6)
PARTNER_2	Alpha-numeric	10	N	Partner ID of the second additional partner. Valid values are in the RMS PARTNER table.	PARTNER_2	VARCHAR2(10)
PARTNER_TYPE_3	Alpha-numeric	6	N	Partner type of the third additional partner. Valid values are in the RMS PARTNER table.	PARTNER_TYPE_3	VARCHAR2(6)
PARTNER_3	Alpha-numeric	10	N	Partner ID of the third additional partner. Valid values are in the RMS PARTNER table.	PARTNER_3	VARCHAR2(10)

Load Scripts

DC_SUPS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_SUPS staging table.

LOAD_SUPS– This function selects from the DC_ SUPS staging table and inserts the data to the RMS SUPS table. All the columns from the staging table defined previously map directly to the RMS table. The following table lists columns that do not exist in the DC_SUPS table and must be defaulted as described.

DC_SUPS to SUPS Column Defaults

Field Name (RMS Table)	Default Value	Comments
SUP_STATUS	A	N/A
AUTO_APPR_DBT_MEMO_IND	Y	If NULL in external table
PREPAY_INVC_IND	Y	N/A
DELIVERY_POLICY	NEXT	If NULL in external table
BRACKET_COSTING_IND	N	If NULL in external table
DSD_IND	N	If NULL in external table
EDI_INVC_IND	Y	N/A
DBT_MEMO_CODE	Y	N/A
INVC_PAY_LOC	C	N/A
INVC_RECEIVE_LOC	C	N/A
ADDINVC_GROSS_NET	G	N/A
VMI_ORDER_STATUS	A	If NULL in external table

Required file to load: dc_sups.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_SUP_ADDR.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_SUP_ADDR staging table.

LOAD_SUP_ADDR– This function selects from the DC_SUP_ADDR staging table and inserts the data to the RMS ADDR table. All the columns from the staging table defined previously map directly to the RMS table. The following table lists columns that do not exist in the DC_SUP_ADDR table and must be defaulted as described.

DC_SUP_ADDRESStoADDRColumnDefaults

Field Name (RMS Table)	Default Value	Comments
ADDR_KEY	Sequence generated	N/A
MODULE	SUPP	N/A
SEQ_NO	1	N/A
ADDR_TYPE	See the note that follows.	N/A

PUBLISH_IND	N	N/A

Note: For each input supplier, the address records are created depending on the mandatory address types in the ADD_TYPE_MODULE table.

Required file to load: dc_sup_addr.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_SUP_IMPORT_ATTR.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_SUP_IMPORT_ATTR staging table.

LOAD_SUP_IMPORT_ATTR– This function selects from the DC_ SUP_IMPORT_ATTR staging table and inserts the data to the RMS SUP_IMPORT_ATTR table. All the columns from the staging Oracle table defined above will directly map to the RMS table.

Required file to load: dc_sup_import_attr.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Post-Loading Requirements

After using the data conversion toolset for this functional area, the SUP_BRACKET_COST table must be loaded manually. This table is required for suppliers that have bracket costing. It must be loaded before you proceed with data conversion for subsequent functional areas, because of data dependencies.

Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_sups.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Partner Overview

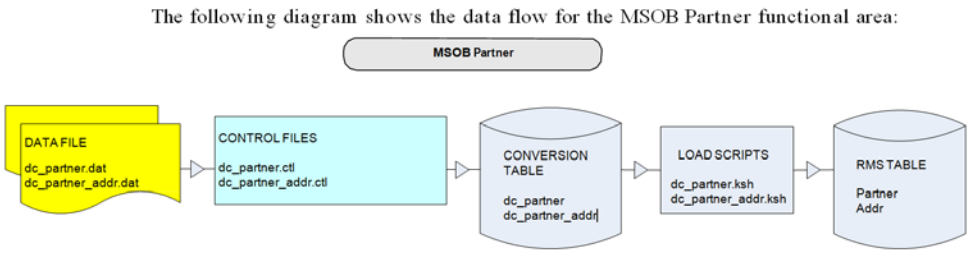
This section describes data conversion for the RMS PARTNER table. The following programs are included in this functional area:

The following programs are included in this functional area:

- Load Scripts
  - dc_partner.ksh
  - dc_partner_addr.ksh
- Control Files:
  - dc_partner.ctl
  - dc_partner_addr.ctl

Data Flow

The following diagram shows the data flow for the MSOB Partner functional area:



Data Flow for the MSOB Partner Functional Area

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_PARTNER Table

File name: **DC_PARTNER.DAT**

Control file: **DC_PARTNER.CTL**

Staging table: **DC_PARTNER**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PARTNER_TYPE	Alpha-numeric	6	Y	Specifies type of partner. E.g. Bank 'BK', etc.	PARTNER_TYPE	VARCHAR2 (6)
PARTNER_ID	Alpha-numeric	10	Y	Unique ID for the partner.	PARTNER_ID	VARCHAR2(10)
PARTNER_DESC	Alpha-numeric	240	Y	Description or name of partner	PARTNER_DESC	VARCHAR2(240)
CURRENCY_CODE	Alpha-numeric	3	Y	Currency for business transaction.	CURRENCY_CODE	VARCHAR2(3)
LANG	Integer	6	N	Partner's preferred language.	LANG	NUMBER(6)
STATUS	Alpha-numeric	1	Y	Determines whether the partner is currently active.	STATUS	VARCHAR2(1)
CONTACT_NAME	Alpha-numeric	120	Y	Name of partner's representative contact.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	Y	Phone number.	CONTACT_PHONE	VARCHAR2(20)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
CONTACT_FAX	Alpha-numeric	20	N	Fax Number.	CONTACT_FAX	VARCHAR2(20)
CONTACT_TELEX	Alpha-numeric	20	N	Telex Number.	CONTACT_TELEX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	Email ID.	CONTACT_EMAIL	VARCHAR2(100)
MFG_ID	Alpha-numeric	18	N	Manufacturer's Tax Identification Number.	MFG_ID	VARCHAR2(18)
PRINCIPLE_COUNTRY_ID	Alpha-numeric	3	N	Country ID to which Partner is assigned.	PRINCIPLE_COUNTRY_ID	VARCHAR2(3)
LINE_OF_CREDIT	Integer	20,4	N	The line of credit the company has at the bank in Partner's currency.	LINE_OF_CREDIT	NUMBER(20,4)
OUTSTAND_CREDIT	Integer	20,4	N	Total amount of credit that the company has used / charged against in the Partner's currency.	OUTSTAND_CREDIT	NUMBER(20,4)
OPEN_CREDIT	Integer	20,4	N	Total amount that the company can still charge against in the Partner's currency.	OPEN_CREDIT	NUMBER(20,4)
YTD_CREDIT	Integer	20,4	N	Total amount of credit the company has used this year to date.	YTD_CREDIT	NUMBER(20,4)
YTD_DRAWDOWNS	Integer	20,4	N	The year to date payments the bank has made on behalf of the company.	YTD_DRAWDOWNS	NUMBER(20,4)
TAX_ID	Alpha-numeric	18	N	Unique Tax Identification Number.	TAX_ID	VARCHAR2(18)
TERMS	Alpha-numeric	15	Y	Payment Terms for partner.	TERMS	VARCHAR2(15)
SERVICE_PERF_REQ_IND	Alpha-numeric	1	Y	Indicates if the expense vendor's services must be confirmed as performed before paying an invoice.	SERVICE_PERF_REQ_IND	VARCHAR2(1)
INVC_PAY_LOC	Alpha-numeric	6	N	Indicates where the invoices from this vendor are paid.	INCV_PAY_LOC	VARCHAR2(6)
INVC_RECEIVE_LOC	Alpha-numeric	6	N	Indicates where the invoices from this vendor are received.	INVC_RECEIVE_LOC	VARCHAR2(6)
IMPORT_COUNTRY_IND	Alpha-numeric	3	N	Import country of the import authority.	IMPORT_COUNTRY_IND	VARCHAR2(3)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRIMARY_IA_IND	Alpha-numeric	1	Y	Indicates whether the import authority is a primary import authority.	PRIMARY_IA_IND	VARCHAR2(1)
COMMENT_DESC	Alpha-numeric	2000	N	Contains any comments associated with partner.	COMMENT_DESC	VARCHAR2(2000)
TSF_ENTITY_ID	Integer	10	N	ID of transfer entity with which External finisher is associated.	TSF_ENTITY_ID	NUMBER(10)
VAT_REGION	Integer	4	N	Vat region with which partner is associated.	VAT_REGION	NUMBER(4)
ORG_UNIT_ID	Integer	15	N	Organization Unit ID.	ORG_UNIT_ID	NUMBER(15)
PARTNER_NAME_SECONDARY	Alpha-numeric	240	N	This will hold the secondary name of the partner.	PARTNER_NAME_SECONDARY	VARCHAR2(240)
AUTO_RCV_STOCK_IND	Alpha-numeric	1	Y	This will indicate whether the system will update the stock for the external finisher when the 1st leg of the transfer is shipped. Valid values are 'Y'es or 'N'o.	AUTO_RECEIVE_IND	VARCHAR2(1)

DC_PARTNER_ADDR Table

File name: **DC_PARTNER_ADDR.DAT**

Control file: **DC_PARTNER_ADDR.CTL**

Staging table: **DC_PARTNER_ADDR**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
KEY_VALUE_1	Alpha-numeric	20	Y	This column contains specific ID or type that the address is attached to. If the module is Partner, then key_value_1 holds the type of Partner [BANK (BK),Freight Forwarder (FF), Factory (FA), Agent (AG), Broker (BR), and Importer (IM)], else it will hold the supplier number.	KEY_VALUE_1	VARCHAR2(20)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
KEY_VALUE_2	Alpha-numeric	20	N	If the module is Partner (PTNR), then this field will contain the partners ID, else this field will be null.	KEY_VALUE_2	VARCHAR2(20)
ADDR_TYPE	Alpha-numeric	2	Y	This column contains a unique number used to distinguish between different addresses.	ADDR_TYPE	VARCHAR2(2)
PRIMARY_ADDR_IND	Alpha-numeric	1	Y	Indicates if this address is the primary for the partner and address type. Valid values are 'Y' (primary) and 'N' (non-primary).	PRIMARY_ADDR_IND	VARCHAR2(1)
CONTACT_NAME	Alpha-numeric	120	N	Contains the name for the primary contact person at this partner address.	CONTACT_NAME	VARCHAR2(120)
CONTACT_PHONE	Alpha-numeric	20	N	Contains the phone number for the contact person at this partner address.	CONTACT_PHONE	VARCHAR2(20)
CONTACT_FAX	Alpha-numeric	20	N	Contains the fax number for this partner address.	CONTACT_FAX	VARCHAR2(20)
CONTACT_EMAIL	Alpha-numeric	100	N	Contains the e-mail address for the contact person at this partner address.	CONTACT_EMAIL	VARCHAR2(100)
CONTACT_TELEX	Alpha-numeric	20	N	Contains the telex number for the contact person at this partner address.	CONTACT_TELEX	VARCHAR2(20)
ADDR_LINE_1	Alpha-numeric	240	Y	Contains the first line of this address for this partner and address type.	ADD_1	VARCHAR2(240)
ADDR_LINE_2	Alpha-numeric	240	N	Contains the second line of this address for this partner and address type.	ADD_2	VARCHAR2(240)
ADDR_LINE_3	Alpha-numeric	240	N	Contains the third line of this address for this partner and address type.	ADD_3	VARCHAR2(240)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
CITY	Alpha-numeric	120	Y	Contains the city of this address for this partner and address type.	CITY	VARCHAR2(120)
COUNTY	Alpha-numeric	250	N	Contains the county of this address for this partner and address type.	COUNTY	VARCHAR2(250)
STATE	Alpha-numeric	3	N	Contains the state of this address for this partner and address type. Values in this column must exist in the RMS STATE table.	STATE	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	Contains the postal code (e.g. Zip Code) of this address for this warehouse and address type.	POST	VARCHAR2(30)
COUNTRY_ID	Alpha-numeric	3	Y	Contains the country code of this address for this warehouse and address type. Values in this column must exist in the RMS COUNTRIES table.	COUNTRY_ID	VARCHAR2(3)
JURISDICTION_CODE	Alpha-numeric	10	N	Jurisdiction code for the address.	JURISDICTION_CODE	VARCHAR2(10)

Load Scripts

DC_PARTNER.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PARTNER staging table.

LOAD_PARTNER– This function contains a PL/SQL block that selects from the DC_PARTNER staging table and inserts the data to the RMS PARTNER table. All the columns from staging table defined previously map directly to the RMS table

The following fields are required:

- PARTNER_ID
- PARTNER_TYPE
- PARTNER_DESC

- CURRENCY_CODE
- STATUS
- CONTACT_NAME
- CONTACT_PHONE
- TERMS
- SERVICE_PERF_REQ_IND
- PRIMARY_IA_IND

Required file to load: dc_partner.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PARTNER_ADDR.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PARTNER_ADDR staging table.

LOAD_PARTNER_ADDR– This function inserts data into the ADDR table by selecting all columns from the DC_PARTNER_ADDR staging table and uses the defaults specified below for the columns that are not in DC_PARTNER_ADDR.

ADDR Column Defaults for partner

Field Name (RMS Table)	Default Value	Comments
ADDR_KEY	system generated	NA
MODULE	PTNR	NA
SEQ_NO	1	NA
PUBLISH_IND	N	NA

Required file to load: dc_partner_addr.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_partner.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

DC_COUNTRY_ATRRIB Table

File name: **DC_COUNTRY_ATTRIB.DAT**

Control File: **DC_COUNTRY_ATTRIB.CTL**

Staging table: **DC_COUNTRY_ATTRIB**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
COUNTRY_ID	Alpha-numeric	3	Y	This contains the code which uniquely identifies the country.	COUNTRY_ID	VARCHAR2 (3)

LOAD SCRIPTS

DC_COUNTRY_ATTRIB.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_COUNTRY_ATTRIB staging table.

LOAD_COUNTRY_ATTRIB– This function selects from the DC_COUNTRY_ATTRIB staging table and inserts the data to the RMS COUNTRY_ATTRIB table. All the columns from the staging oracle table defined above will directly map to the RMS table. The table below lists columns that do not exist on DC_ COUNTRY_ATTRIB and will need to be defaulted as described.

Required file to load: dc_country_attrib.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_country_attrib.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

.

7

Items

Because different types of items have different data structures, the Items functional area is organized based on item types, as follows:

- Fashion Items
- Hardline Items
- Grocery Items
- Pack Items
- Item Supplier
- Item Location
- Others

Note the following:

- Break-to-sell items are not supported in this data conversion toolset.
- 2- to 3-tier non-pack items are both orderable and sellable.
- Pack items are divided into sellable only and orderable (sellable is optional).

Prerequisites

Before you begin using the data conversion toolset for Items, you must complete data conversion for the following functional areas:

- Core
- Merchandise Hierarchy
- Organizational Hierarchy
- Suppliers

There are tables that must be loaded manually, because of data dependencies for auto-loading within this functional area. Manual data loading can be done online through Merchandising applications (RMS, RPM), or scripts can be created. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs. Fashion

Items Overview

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- ITEM_MASTER
- VAT_ITEM (only if system_optinos.vat_ind is Y and default_tax_type is not GTAX)
- UDA_ITEM_LOV
- ITEM_CHRG_HEAD
- ITEM_CHRG_DETAIL

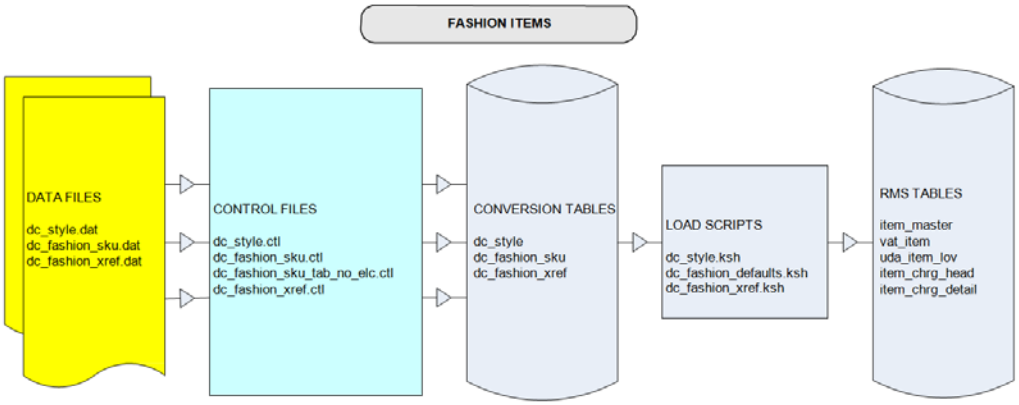
The following programs are included in this functional area:

- Load Scripts:
  - dc_style.ksh
  - dc_fashion_xref.ksh

  - dc_fashion_defaults.ksh
- Control Files:
  - dc_style.ctl
  - dc_fashion_sku.ctl
  - dc_fashion_sku_tab_no_elc.ctl
  - dc_fashion_xref.ctl

Data Flow

The following diagram shows the data flow for the Fashion Items functional area:



Data Flow for the Fashion Items Functional Area

Prerequisites

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_STYLE Table

File name: **DC_STYLE.DAT**

Control file: **DC_STYLE.CTL**

Staging table: **DC_STYLE**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL < ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.PACK_IND = N, and compare to flat file DC_STYLE.DAT to ensure that all rows are loaded.
- Ensure that ITEM_MASTER.DEPT/ITEM_MASTER.CLASS/ ITEM_MASTER.SUBCLASS combination exists in SUBCLASS.
- Ensure that ITEM_MASTER.DIFF_1 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_2 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_3 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_4 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
STYLE	Alpha-numeric	20	Y	ID that uniquely identifies the style.	ITEM	VARCHAR2(25)
STYLE_DESC	Alpha-numeric	250	Y	Description of the style.	ITEM_DESC	VARCHAR2(250)
STYLE_SHORT_DESC	Alpha-numeric	120	N	Short description of the style.Default = First 120 characters of SKU_DESC.	SHORT_DESC	VARCHAR2(120)
STYLE_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the item for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 of which merchandise hierarchy level 5 is a member. Valid values are in the DEPT field in the DEPS table in RMS.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifier of the merchandise hierarchy level 5 which is a member of merchandise hierarchy level 4. Valid values are in the CLASS field in the CLASS table in RMS.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Identifier of the merchandise hierarchy level 6 which is a member of merchandise hierarchy level 5. Valid values are in the SUBCLASS field in the SUBCLASS table in RMS.	SUBCLASS	NUMBER(4)
SIZE_1_GROUP	Alpha-numeric	10	Y	Size group ID of the first size that differentiates the style from its ITEM_PARENT (for example, men's pant sizes or a value such as 6 oz). Valid values are in the DIFF_GROUP and DIFF_IDS tables.	SIZE_1_GROUP	VARCHAR2(10)
SIZE_2_GROUP	Alpha-numeric	10	N	Size group ID of the second size that differentiates the style from its ITEM_PARENT. Valid values are in the DIFF_GROUP and DIFF_IDS tables.	SIZE_2_GROUP	VARCHAR2(10)
COLOR_GROUP	Alpha-numeric	10	N	ID of the color grouping of the style that differentiates the style from its ITEM_PARENT (for example, pastel colors). Valid values are in the DIFF_GROUP and DIFF_IDS tables.	COLOR_GROUP	VARCHAR2(10)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
OTHER_DIFF_GROUP	Alpha-numeric	10	N	ID of the other grouping of the style that differentiates the style from its ITEM_PARENT. Valid values are in the DIFF_GROUP and DIFF_IDS tables.	OTHER_GROUP	VARCHAR2(10)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
ITEM_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings, such as style/color, is achieved by adding this indicator for each grouping in the table. This item aggregate indicator allows the user to specify whether the item can aggregate by numbers. Aggregation allows the system to support allocations at a style/grouping level. The remainder of the diffs that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	ITEM_AGGREGATE	VARCHAR2(1)
SIZE_1_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings, such as style/color, is achieved by adding this indicator for each grouping in the table. This item aggregate indicator allows the user to specify whether the item can aggregate by numbers. Aggregation allows the system to support allocations at a style/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	SIZE_1_AGGREGATE	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SIZE_2_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings, such as style/color, is achieved by adding this indicator for each grouping in the table. This item aggregate indicator allows the user to specify whether the item can aggregate by numbers. Aggregation allows the system to support allocations at a style/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	SIZE_2_AGGREGATE	VARCHAR2(1)
COLOR_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings, such as style/color, is achieved by adding this indicator for each grouping in the table. This item aggregate indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a style/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	COLOR_AGGREGATE	VARCHAR2(1)
OTHER_DIFF_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings, such as style/color, is achieved by adding this indicator for each grouping in the table. This item aggregate indicator allows the user to specify whether the item can aggregate by numbers. Aggregation allows the system to support allocations at a style/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	OTHER_AGGREGATE	VARCHAR2(1)
STYLE_COMMENTS	Alpha-numeric	2000	N	Comments associated with the style.	STYLE_COMMENTS	VARCHAR2(2000)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PERISHABLE_IND	Alpha-numeric	1	N	A grocery item attribute used to indicate whether an item is perishable or not.	PERISHABLE_IND	VARCHAR2(1)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

Note: The same number of aggregate indicators should be populated as the number of corresponding diff values.

For example, if diffs 1 and 2 contain values, then only diff aggregate 1 and diff aggregate 2 should be populated with a Y or N. The diff 3 and diff 4 aggregate indicators should be NULL.

For item aggregation, the item can aggregate only by up to 1 less than the total number of differentiator groups specified. For example, if an item has three differentiator groups associated with it, the user can aggregate by as many as two of those groups.

DC_FASHION_SKU Table

File name: **DC_FASHION_SKU.DAT**

Control file: **DC_FASHION_SKU.CTL**

Staging table: **DC_FASHION_SKU**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL = ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.PACK_IND = N, and compare to flat file DC_FASHION_SKU.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SKU	Alpha-numeric	20	Y	ID that uniquely identifies the stock keeping unit.	ITEM	VARCHAR2(25)
PRIMARY_SKU_IND	Alpha-numeric	1	Y	Not in RMS ITEM_MASTER, needed for defaulting style.	PRIMARY_SKU_IND	VARCHAR2(1)
STYLE	Alpha-numeric	20	Y	Style associated with the SKU.	ITEM_PARENT	VARCHAR2(25)
SKU_DESC	Alpha-numeric	250	Y	Description of the SKU.	ITEM_DESC	VARCHAR2(250)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SHORT_DESC	Alpha-numeric	120	Y	Short description of the SKU. Default = First 120 characters of SKU_DESC	SHORT_DESC	VARCHAR2(120)
SKU_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
COST_ZONE_GROUP_ID	Integer	-	Y	Cost zone group associated with the item. This field is only required when ELC_IND (landed cost indicator) is set to Y in the SYSTEM_OPTIONS table.	COST_ZONE_GROUP_ID	NUMBER(4)
STANDARD_UOM	Alpha-numeric	4	Y	Unit of measure in which stock of the item is tracked at a corporate level.	STANDARD_UOM	VARCHAR2(4)
UOM_CONV_FACTOR	Numeric	12,10	N	Conversion factor between an each and the STANDARD_UOM, when the STANDARD_UOM is not in the quantity class (for example, if STANDARD_UOM = lb and 1 lb = 10 eaches, this factor is 10). This factor is used to convert sales and stock data when an item is retailed in eaches, but does not have eaches as its standard unit of measure.	UOM_CONV_FACTOR	NUMBER(20,10)
STORE_ORDER_MULT	Alpha-numeric	1	Y	Unit type in which merchandise shipped from the warehouses to the stores must be specified. Valid values are: C - Cases I - Inner E - Eaches	STORE_ORDER_MULT	VARCHAR2(1)
SKU_COMMENTS	Alpha-numeric	2000	N	Comments associated with the SKU.	SKU_COMMENTS	VARCHAR2(2000)
MERCHANDISE_IND	Alpha-numeric	1	N	Indicates if the item is a merchandise item (Y, N). Default = Y	MERCHANDISE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
FORECAST_IND	Alpha-numeric	1	N	Indicates if this item will be interfaced to an external forecasting system (Y, N). Default = Y	FORECAST_IND	VARCHAR2(1)
SIZE_1	Alpha-numeric	10	N	Size ID of the first size that differentiates the SKU from its Style (for example, 34 waist). Valid values are in the DIFF_GROUP and DIFF_ID tables.	SIZE_1	VARCHAR2(10)
SIZE_2	Alpha-numeric	10	N	Size ID of the first size that differentiates the SKU from its style (for example, 32 length). Valid values are in the DIFF_GROUP and DIFF_ID tables.	SIZE_2	VARCHAR2(10)
COLOR	Alpha-numeric	10	N	Color ID of the color that differentiates the SKU from its style (for example, red). Valid values are found in the DIFF_GROUP and DIFF_ID tables.	COLOR	VARCHAR2(10)
OTHER_VARIANT	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its style (for example, stone-washed). Valid values are in the DIFF_GROUP and DIFF_ID tables.	OTHER_VARIANT	VARCHAR2(10)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
PERISHABLE_IND	Alpha-numeric	1	N	A grocery item attribute used to indicate whether an item is perishable or not.	PERISHABLE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

DC_FASHION_XREF Table

File name: **DC_FASHION_XREF.DAT**

Control file: **DC_FASHION_XREF.CTL**

Staging table: **DC_FASHION_XREF**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL > ITEM_MASTER.TRAN_LEVEL, and compare to flat file DC_FASHION_XREF.DAT to ensure that all rows are loaded.
- Ensure that ITEM_MASTER.ITEM is unique.
- Ensure that ITEM_MASTER.ITEM_PARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the child less 1.
- Ensure that ITEM_MASTER.ITEM_GRANDPARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the grandchild less 2.
- Ensure that ITEM_MASTER.COST_ZONE_GROUP_ID is a valid COST_ZONE_GROUP..ZONE_GROUP_ID if SYSTEM_OPTIONS.ELC_IND = Y.
- Ensure that ITEM_MASTER.STANDARD_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS is not MISC.
- Ensure that ITEM_MASTER.UOM_CONV_FACTOR is not NULL if UOM_CLASS of ITEM_MASTER.STANDARD_UOM is not QTY.
- Ensure that ITEM_MASTER.PACKAGE_UOM (if not NULL) is a valid UOM_CLASS.UOM.
- Ensure that ITEM_MASTER.RETAIL_LABEL_TYPE (if not NULL) is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = RTLT.
- Ensure that ITEM_MASTER.HANDLING_TEMP (if not NULL) is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = HTMP.
- Ensure that ITEM_MASTER.HANDLING_SENSITIVITY (if not NULL) is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = HSEN.
- Ensure that ITEM_ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = UPCT.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
XREF_ITEM	Alpha-numeric	25	Y	The ID that uniquely identifies the scanning barcode associated with a product.	ITEM	VARCHAR2(25)
XREF_ DESC	Alpha-numeric	250	Y	Description of the item.	ITEM_DESC	VARCHAR2(250)
XREF_SHORT_DESC	Alpha-numeric	120	N	Default = First 120 characters of xref_desc.	SHORT_DESC	VARCHAR2(120)
XREF_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
SKU	Alpha-numeric	25	Y	Stock keeping unit associated with the xref_item.	ITEM_PARENT	VARCHAR2(25)
STYLE	Alpha-numeric	25	Y	Style associated with the xref_item.	ITEM_GRANDPARENT	VARCHAR2(25)
XREF_COMMENTS	Alpha-numeric	2000	N	Comments associated with the xref_item.	STYLE_COMMENTS	VARCHAR2(2000)
PRIMARY_REF_ITEM_IND	Alpha-numeric	1	N	Indicates that xref_item is the primary item for the stock keeping unit. Note – there can only be one primary xref item for a SKU. Default = N.	PRIMARY_REF_IND	VARCHAR2(1)
ITEM_NUMBER_TYPE	Alpha-numeric	6	Y	Code specifying what type the xref_item is. Valid values for this field are in the code type UPCT on the code_head and code_detail tables. Examples are UPC, EAN and others.	ITEM_NUMBER_TYPE	VARCHAR2(6)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_ TYPE	VARCHAR2(6)
PERISHABLE_IND	Alpha-numeric	1	N	A grocery item attribute used to indicate whether an item is perishable or not.	PERISHABLE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

Load Scripts

DC_STYLE.ksh

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_STYLE_FILE – This function call SQLLOADER to load data from input file to DC_STYLE staging table.

LOAD_FASHION_FILE – This function call SQLLOADER to load data from input file to DC_FASHION_SKU staging table.

LOAD_STYLE_SKU– This function contains a PL/SQL block that selects from the DC_STYLE and the DC_FASHION_SKU staging tables and inserts the data to the RMS ITEM_MASTER table.

Styles

For styles, the following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_STYLE and DC_FASHION_SKU to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	1	NA
TRAN_LEVEL	2	NA
SHORT_DESC	SUBSTR 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA

Column Name (RMS Table)	Default Value	Comments
STATUS	A	NA
CREATE_DATETIME	SYSDATE	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	SYSDATE	NA
ITEM_AGGREGATE_IND	N	If NULL
DIFF_1_AGGREGATE_IND	N	If NULL
DIFF_2_AGGREGATE_IND	N	If NULL
DIFF_3_AGGREGATE_IND	N	If NULL
DIFF_4_AGGREGATE_IND	N	If NULL
PERISHABLE_IND	N	N/A

SKUs

For SKUs, the following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_FASHION_SKU to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	2	NA
TRAN_LEVEL	2	NA
SHORT_DESC	SUBSTR 120 characters from SKU_DESC	If NULL
DESC_UP	Upper	NA
STATUS	A	NA
CREATE_DATETIME	SYSDATE	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
ORDERABLE_IND	Y	NA
SELLABLE_IND	Y	NA
MERCHANDISE_IND	Y	If NULL
FORECAST_IND	Y	If NULL
INVENTORY_IND	Y	NA
ITEM_AGGREGATE_IND	N	NA

Column Name (RMS Table)	Default Value	Comments
DIFF_1_AGGREGATE_IND	N	NA
DIFF_2_AGGREGATE_IND	N	NA
DIFF_3_AGGREGATE_IND	N	NA
DIFF_4_AGGREGATE_IND	N	NA
PRIMARY_REF_ITEM_IND	N	NA
CONST_DIMEN_IND	N	NA
GIFT_WRAP_IND	N	NA
SHIP_ALONE_IND	N	NA
ITEM_XFORM_IND	N	NA
PACK_IND	N	NA
SIMPLE_PACK_IND	N	NA
CATCH_WEIGHT_IND	N	NA
CONTAINS_INNER_IND	N	NA
PERISHABLE_IND	N	NA

Required file to load: dc_style.dat, dc_fashion_sku.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_FASHION_DEFAULTS.KSH

This ksh script will be called to insert item defaults from merchandise hierarchy specification.

The following functions are defined in the declaration of the script:

LOAD_FASHION_DEFAULTS– This function inserts item defaults from the merchandise hierarchy specifications for VAT,UDAs and ITEM Charges. Create two cursors to retrieve using bulk collect into a PL/SQL table the ITEM, DEPT, CLASS and SUBCLASS values from DC_STYLE and from DC_STYLE joined with DC_FASHION_SKU.

If vat is turned on in system_options and default tax type is not GTAX (SVAT is used), Retrieve SKU information and call the VAT_SQL.DEFAULT_VAT_ITEM. Retrieve style information and call UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL.DC_DEFAULT_CHRGS. Retrieve sku information and call UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL.DEFAULT_CHRGS.

Required file to load: dc_style.dat and dc_fashion_sku.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and.

COMMIT: Follow each insert statement with a commit command.

DC_FASHION_XREF.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions are defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_FASHION_XREF staging table.

LOAD_FASHION_XREF– This function contains a PL/SQL block that selects from the DC_FASHION_XREF and the DC_FASHION_SKU staging tables and inserts the data to the RMS ITEM_MASTER table.

Most of the columns from the staging table defined above directly map to the RMS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_FASHION_XREF and DC_FASHION_SKU to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_LEVEL	3	NA
TRAN_LEVEL	2	NA
SHORT_DESC	SUBSTR 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
PRIMARY_REF_ITEM_IND	N	If NULL
CREATE_DATETIME	SYSDATE	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	SYSDATE	NA
PERISHABLE_IND	N	NA

Required file to load: dc_style.dat, dc_fashion_sku.dat and dc_fashion_xref.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts are executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_style.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Hardlines Items Overview

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- ITEM_MASTER
- VAT_ITEM (only if system_optinos.vat_ind is Y and default_tax_type is not GTAX)
- UDA_ITEM_LOV
- ITEM_CHRG_HEAD
- ITEM_CHRG_DETAIL

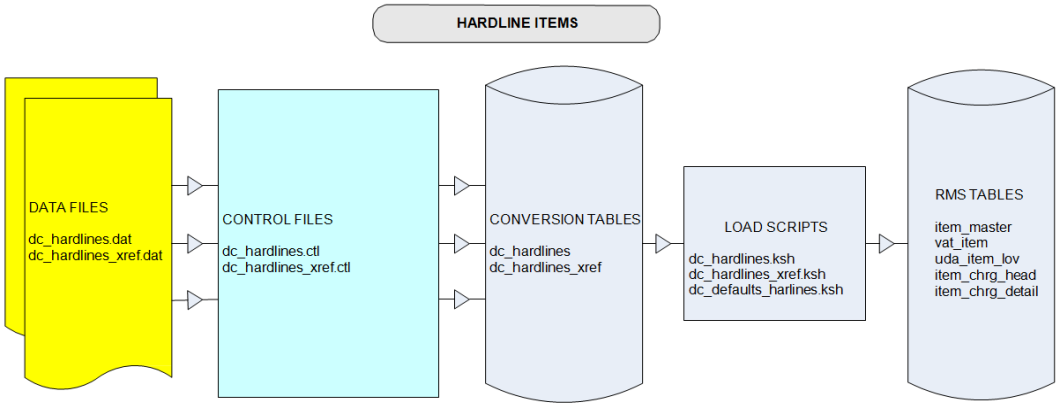
The following programs are included in this functional area.

- Load Scripts:
  - dc_hardlines.ksh
  - dc_hardlines_xref.ksh
  - dc_default_hardlines.ksh
- Control Files:
  - dc_hardlines.ctl
  - dc_hardlines_xref.ctl

Data Flow

The following diagram shows the data flow for the Hardline Items functional area.

The following diagram shows the data flow for the Hardline Items functional area.



Data Flow for the Hardline Functional Area

Prerequisites

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_HARDLINES Table

File name: **DC_HARDLINES.DAT**

Control file: **DC_HARDLINES.CTL**

Staging table: **DC_HARDLINES**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL = ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.ITEM_PARENT is NULL and

ITEM_MASTER.PACK_IND = N, and compare to flat file DC_HARDLINES.DAT to ensure that all rows are loaded.

- Ensure that ITEM_MASTER.DEPT/ITEM_MASTER.CLASS/ ITEM_MASTER.SUBCLASS combination exists in SUBCLASS.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
SKU	Alpha-numeric	20	Y	ID that uniquely identifies the stock keeping unit.	ITEM	VARCHAR2(25)
SKU_DESC	Alpha-numeric	120	Y	Description of the SKU.	ITEM_DESC	VARCHAR2(250)
SKU_SHORT_DESC	Alpha-numeric		N	Short description of the SKU. Default = First 120 characters of SKU_DESC.	SHORT_DESC	VARCHAR2(120)
SKU_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 of which merchandise hierarchy level 5 is a member. Valid values are in the DEPT field in the DEPS table in RMS.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifier of the merchandise hierarchy level 5 which is a member of merchandise hierarchy level 4. Valid values are in the CLASS field in the CLASS table in RMS.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Identifier of the merchandise hierarchy level 6 which is a member of merchandise hierarchy level 5. Valid values are in the SUBCLASS field in the SUBCLASS table in RMS.	SUBCLASS	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
COST_ZONE_GROUP_ID	Integer		Y	Cost zone group associated with the item. This field is only required when ELC_IND (landed cost indicator) is set to Y in the SYSTEM_OPTIONS table.	COST_ZONE_GROUP_ID	NUMBER(4)
UOM_CONV_FACTOR	Floating Point	12,10	N	Conversion factor between an each and the STANDARD_UOM, when the STANDARD_UOM is not in the quantity class. (For example, if STANDARD_UOM = lb and 1 lb = 10 eaches, this factor is 10). This factor is used to convert sales and stock data when an item is retailed in eaches, but does not have eaches as its standard unit of measure.	UOM_CONV_FACTOR	NUMBER(20,10)
STANDARD_UOM	Alpha-numeric	4	Y	Unit of measure in which stock of the item is tracked at a corporate level.	STANDARD_UOM	VARCHAR2(4)
STORE_ORDER_MULT	Alpha-numeric	1	Y	Unit type in which merchandise shipped from the warehouses to the stores must be specified. Valid values are: C - Cases I - Inner E - Eaches	STORE_ORD_MULT	VARCHAR2(1)
SKU_COMMENTS	Alpha-numeric	2000	N	Comments associated with the SKU.	COMMENTS	VARCHAR2(2000)
MERCHANDISE_IND	Alpha-numeric	1	N	Indicates whether the item is a merchandise item (Y or N). Default = Y.	MERCHANDISE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
FORECAST_IND	Alpha-numeric	1	N	Indicates whether this item will be interfaced to an external forecasting system (Y or N). Default = Y.	FORECAST_IND	VARCHAR2(1)

DC_HARDLINES_XREF Table

File name: **DC_HARDLINES_XREF.DAT**

Control file: **DC_HARDLINES_XREF.CTL**

Staging table: **DC_HARDLINES_XREF**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL > ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.ITEM_GRANDPARENT is NULL, and compare to flat file DC_HARDLINES_XREF.DAT to ensure that all rows are loaded.
- Ensure that ITEM_MASTER.ITEM is unique.
- Ensure that ITEM_MASTER.ITEM_PARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the child less 1.
- Ensure that ITEM_MASTER.COST_ZONE_GROUP_ID is a valid COST_ZONE_GROUP. ZONE_GROUP_ID if SYSTEM_OPTIONS.ELC_IND = Y.
- Ensure that ITEM_MASTER.STANDARD_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS is not MISC.
- Ensure that ITEM_MASTER.UOM_CONV_FACTOR is not NULL if UOM_CLASS of ITEM_MASTER.STANDARD_UOM is not QTY.
- Ensure that ITEM_ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = UPCT.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
XREF_ITEM	Alpha-numeric	25	Y	ID that uniquely identifies the scanning bar code associated with a product.	ITEM	VARCHAR2(25)
XREF_DESC	Alpha-numeric	250	Y	Description of the item.	ITEM_DESC	VARCHAR2(250)
XREF_SHORT_DESC	Alpha-numeric	120	N	Default = 120 characters of XREF_DESC.	SHORT_DESC	VARCHAR2(120)
XREF_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
SKU	Alpha-numeric	25	Y	Stock keeping unit associated with the XREF_ITEM.	ITEM_PARENT	VARCHAR2(25)
XREF_COMMENTS	Alpha-numeric	2000	N	Comments associated with the XREF_ITEM.	COMMENTS	VARCHAR2 (2000)
PRIMARY_REF_ITEM_IND	Alpha-numeric	1	N	Indicates whether XREF_ITEM is the primary item for the stock keeping unit. **Note:** There can only be one primary xref item for a SKU. Default = N	PRIMARY_REF_ITEM_IND	VARCHAR2(1)
ITEM_NUMBER_TYPE	Alpha-numeric	6	Y	Code specifying what type the XREF_ITEM is. Valid values for this field are in the code type UPCT in the CODE_HEAD and CODE_DETAIL tables.	ITEM_NUMBER_TYPE	VARCHAR2(6)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)

Load Scripts

DC_HARDLINES.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_HARDLINES staging table.

LOAD_HARDLINES– This function contains a PL/SQL block that selects from the DC_HARDLINES staging table and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_HARDLINES to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	1	NA
TRAN_LEVEL	1	NA
SHORT_DESC	rtrim of substr b 120 characters from ITEM_DESC.	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
ORDERABLE_IND	Y	NA
SELLABLE_IND	Y	NA
INVENTORY_IND	Y	NA
MERCHANDISE_IND	Y	If NULL
FORECAST_IND	Y	If NULL
ITEM_AGGREGATE_IND	N	NA
DIFF_1_AGGREGATE_IND	N	NA
DIFF_2_AGGREGATE_IND	N	NA
DIFF_3_AGGREGATE_IND	N	NA
DIFF_4_AGGREGATE_IND	N	NA

Column Name (RMS Table)	Default Value	Comments
PRIMARY_REF_ITEM_IND	N	NA
CONST_DIMEN_IND	N	NA
GIFT_WRAP_IND	N	NA
SHIP_ALONE_IND	N	NA
ITEM_XFORM_IND	N	NA
PACK_IND	N	NA
SIMPLE_PACK_IND	N	NA
CATCH_WEIGHT_IND	N	NA
CONTAINS_INNER_IND	N	NA
PERISHABLE_IND	N	NA

Required file to load: dc_hardlines.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_DEFAULT_HARDLINES.KSH

This ksh script will be called to insert item defaults from the merchandise hierarchy specifications for VAT,UDAs and ITEM Charge.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_DEFAULTS– This function inserts item defaults from the merchandise hierarchy specifications for VAT, UDAs, and item charges. Using bulk collect, it retrieves into a PL/SQL table the ITEM, DEPT, CLASS, and SUBCLASS values from the DC_HARDLINES table.

If the VAT indicator is turned on in SYSTEM_OPTIONS and default_tax_type is NOT GTAX (i.e. SVAT is used), this function retrieves SKU information and calls the VAT_SQL.DEFAULT_VAT_ITEM to default data into the RMS VAT_ITEM table.

It retrieves item information and calls UDA_SQL.INSERT_DEFAULTS and

ITEM_CHARGE_SQL.DC_DEFAULT_CHRGS. These functions default data into the RMS UDA_ITEM_LOV, ITEM_CHRG_HEAD, and ITEM_CHRG_DETAIL tables.

Required file to load: dc_hardlines.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_HARDLINES_XREF.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_HARDLINES_XREF staging table.

LOAD_HARDLINES_XREF– This function contains a PL/SQL block that selects from the DC_HARDLINES_XREF staging tables and inserts the data to the RMS ITEM_MASTER table.

Most of the columns from the staging table defined above map directly to the RMS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_XREF to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_LEVEL	2	NA
TRAN_LEVEL	1	NA
SHORT_DESC	rtrim of substr b 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
PRIMARY_REF_ITEM_IND	N	If NULL
PERISHABLE_IND	N	NA

Required file to load: dc_hardlines_xref.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_hardlines.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Grocery Items Overview

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

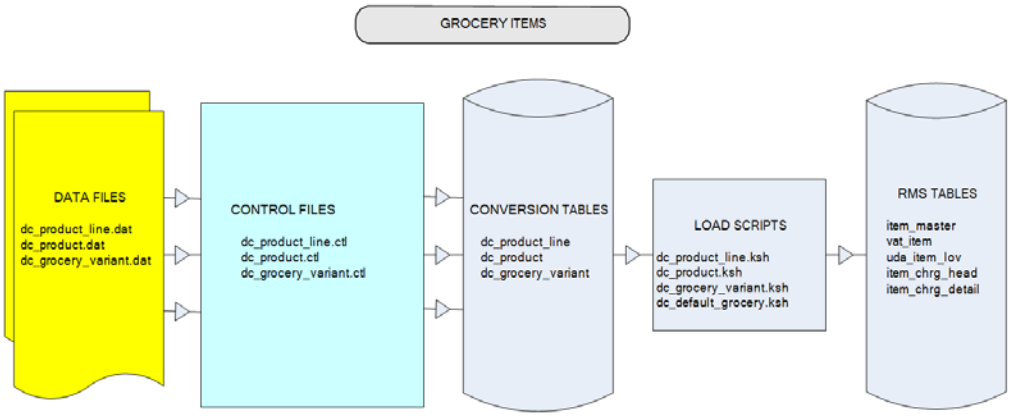
- ITEM_MASTER
- VAT_ITEM (only if system_optinos.vat_ind is Y and default_tax_type is not GTAX)
- UDA_ITEM_LOV
- ITEM_CHRG_HEAD
- ITEM_CHRG_DETAIL

The following programs are included in this functional area:

- Load Scripts:
  - dc_product_line.ksh
  - dc_product.ksh
  - dc_grocery_variant.ksh
  - dc_default_grocery.ksh
- Control Files:
  - dc_product_line.ctl
  - dc_product.ctl
  - dc_grocery_variant.ctl

Data Flow

The following diagram shows the data flow for the Grocery Items functional area:



Data Flow for the Grocery Items Functional Area

Prerequisites

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_PRODUCT_LINE Table

File name: **DC_PRODUCT_LINE.DAT**

Control file: **DC_PRODUCT_LINE.CTL**

Staging table: **DC_PRODUCT_LINE**

Suggested post-loading validation:

- Ensure that ITEM_MASTER.DEPT/ITEM_MASTER.CLASS/ ITEM_MASTER.SUBCLASS combination exists in SUBCLASS.

- Ensure that ITEM_MASTER.DIFF_1 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_2 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_3 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_4 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRODUCT_LINE	Alpha-numeric	25	Y	ID that uniquely identifies the product line.	ITEM	VARCHAR2(25)
PRODUCT_LINE_DESC	Alpha-numeric	250	Y	Description of the product line.	ITEM_DESC	VARCHAR2(250)
PRODUCT_LINE_SHORT_DESC	Alpha-numeric	120	N	Short description of the product line. Default = First 120 characters of ITEM_DESC	SHORT_DESC	VARCHAR2(120)
PRODUCT_LINE_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of product line.	ITEM_DESC_SECONDARY	VARCHAR2(250)
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 of which merchandise hierarchy level 5 is a member. Valid values are in the DEPT field in the DEPS table in RMS.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifies the merchandise hierarchy level 5 which is a member of merchandise hierarchy level 4. Valid values are in the CLASS field in the CLASS table in RMS.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Identifies the merchandise hierarchy level 6 which is a member of merchandise hierarchy level 5. Valid values are in the SUBCLASS field in the SUBCLASS table in RMS.	SUBCLASS	NUMBER(4)
DIFF_GROUP_1_FLAVOR	Alpha-numeric	10	N	Flavor group ID that differentiates the product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_1	VARCHAR2(10)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
DIFF_GROUP_2_SIZE	Alpha-numeric	10	N	Size group ID that differentiates the product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_2	VARCHAR2(10)
OTHER_GROUP_3	Alpha-numeric	10	N	ID of a grouping that differentiates the product line. Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_3	VARCHAR2(10)
OTHER_GROUP_4	Alpha-numeric	10	N	ID of a grouping that differentiates the product line. Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_4	VARCHAR2(10)
ITEM_AGGREGATE	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to specific groupings such as product line/flavor. This item aggregate indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a product line/grouping level. The differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	ITEM_AGGREGATE_IND	VARCHAR2(1)
FLAVOR_AGGREGATE_IND	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to a product line/flavor level. This indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a product line/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	DIFF_1_AGGREGATE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SIZE_AGGREGATE_IND	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to a product line/size grouping level. This indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a product line/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	DIFF_2_AGGREGATE_IND	VARCHAR2(1)
OTHER_GROUP_3__AGGREGATE_IND	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to a product line/other grouping level. This indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a product line/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	DIFF_3_AGGREGATE_IND	VARCHAR2(1)
OTHER_GROUP_4_AGGREGATE_IND	Alpha-numeric	1	N	Default = N Indicator for the item aggregating up to a product line/grouping level. This indicator allows the user to specify if the item may aggregate by numbers. Aggregation allows the system to support allocations at a product line/grouping level. The remaining differentiators that are not a part of the aggregate group represent the curve portion of the allocation algorithm.	DIFF_4_AGGREGATE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRODUCT_LINE_COMMENTS	Alpha-numeric	2000	N	Comments associated with the product line.	COMMENTS	VARCHAR2(2000)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

Note: The same number of aggregate indicators should be populated as the number of corresponding differentiator values.

For example, if Diffs 1 and 2 contain values, then only diff aggregate 1 and diff aggregate 2 should be populated with a Y or N. The diff 3 and 4 aggregate indicators should be NULL.

For item aggregation, the item can only aggregate by up to 1 less than the total number of differentiator groups specified. For example, if an item has three differentiator groups associated with it, the user can aggregate by as many as two of those groups.

DC_PRODUCT Table

File name: **DC_PRODUCT.DAT**

Control file: **DC_PRODUCT.CTL**

Staging table: **DC_PRODUCT**

Separate post-loading validation is not required for the DC_PRODUCT table. The validations for the DC_GROCERY_VARIANT table (later in this chapter) will also validate the rows loaded to the DC_PRODUCT table.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRODUCT	Alpha-numeric	25	Y	ID that identifies the product.	ITEM	VARCHAR2(25)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PRIMARY_PRODUCT_IND	Alpha-numeric	1	N	Not in the RMS ITEM_MASTER table, needed for defaulting product line.	PRIMARY_PRODUCT_IND	VARCHAR2(1)
PRODUCT_LINE	Alpha-numeric	25	Y	Product line associated with the product.	ITEM_PARENT	VARCHAR2(25)
PRODUCT_DESC	Alpha-numeric	250	Y	Description of the product.	ITEM_DESC	VARCHAR2(250)
PRODUCT_SHORT_DESC	Alpha-numeric	120	N	Default = First 120 characters of ITEM_DESC (PRODUCT_DESC)	SHORT_DESC	VARCHAR2(120)
PRODUCT_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the product.	ITEM_DESC_SECONDARY	VARCHAR2(250)
COST_ZONE_GROUP_ID	Integer	4	Y	Cost zone group associated with the product. This field is only required when ELC_IND (landed cost indicator) is set to Y in the SYSTEM_OPTIONS table within RMS.	COST_ZONE_GROUP_ID	NUMBER(4)
UOM_CONV_FACTOR	Numeric	20,10	N	Conversion factor between an each and the STANDARD_UOM when the STANDARD_UOM is not in the quantity class. (For example, if STANDARD_UOM = lb and 1 lb = 10 eaches, this factor is 10). This factor is used to convert sales and stock data when an item is retailed in eaches, but does not have eaches as its standard unit of measure.	UOM_CONV_FACTOR	NUMBER(20,10)
STANDARD_UOM	Alpha-numeric	4	Y	Unit of measure in which stock of the product is tracked at a corporate level.	STANDARD_UOM	VARCHAR2(4)
STORE_ORD_MULT	Alpha-numeric	1	Y	Unit type in which products shipped from the warehouses to the stores must be specified. Valid values are: C - Cases I - Inner E - Eaches	STORE_ORD_MULT	VARCHAR2(1)
MERCHANDISE_IND	Alpha-numeric	1	N	Default = Y Indicates if the product is a merchandise item (Y or N).	MERCHANDISE_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
FORECAST_IND	Alpha-numeric	1	N	Default = Y Indicates if this product will be interfaced to an external forecasting system (Y or N).	FORECAST_IND	VARCHAR2(1)
DIFF_1_FLAVOR	Alpha-numeric	10	N	Flavor ID of the flavor that differentiates the product from its product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_1	VARCHAR2(10)
DIFF_2_SIZE	Alpha-numeric	10	N	Size ID of the size that differentiates the product from its product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_2	VARCHAR2(10)
OTHER_DIFF_3	Alpha-numeric	10	N	ID of the differentiator that differentiates the product from its product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_3	VARCHAR2(10)
OTHER DIFF_4	Alpha-numeric	10	N	ID of the differentiator that differentiates the product from its product line. Valid values are in the DIFF_GROUP and DIFF_IDS tables in RMS.	DIFF_4	VARCHAR2(10)
CATCH_WEIGHT_IND	Alpha-numeric	1	N	Default = N Indicates whether the item should be weighed when it arrives at a location. Valid values for this field are Y and N.	CATCH_WEIGHT_IND	VARCHAR2(1)
HANDLING_TEMP	Alpha-numeric	6	N	Temperature information associated with the item. Valid values for this field are in the code type HTMP in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_TEMP	VARCHAR2(6)
HANDLING_SENSITIVITY	Alpha-numeric	6	N	Sensitivity information associated with the item. Valid values for this field are in the code type HSEN in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_SENSITIVITY	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
WASTE_TYPE	Alpha-numeric	6	N	Identifies wastage type as either sales or spoilage wastage. Sales wastage occurs during processes that make an item saleable (for example, fat is trimmed off at customer request). Spoilage wastage occurs during the product's shelf life (for example, evaporation causes the product to weigh less after a period of time). Valid values are: SP - Spoilage SL - Sales Wastage is not applicable to pack items.	WASTE_TYPE	VARCHAR2(6)
WASTE_PCT	Alpha-numeric	12,4	N	Average percent of wastage for the item over its shelf life. Used in inflating the retail price for wastage items.	WASTE_PCT	NUMBER(12,4)
DEFAULT_ WASTE_PCT	Alpha-numeric	12,4	N	Default daily wastage percent for spoilage type wastage items. This value defaults to all item locations and represents the average amount of wastage that occurs on a daily basis.	DEFAULT_ WASTE_PCT	NUMBER(12,4)
PACKAGE_ SIZE	Alpha-numeric	12,4	N	Size of the product printed on any packaging (for example, 24 ounces). This field is used for reporting purposes, as well as by Retail Price Management to determine same-sized and different-sized items.	PACKAGE_SIZE	NUMBER(12,4)
PACKAGE_ UOM	Alpha-numeric	4	N	Unit of measure associated with the package size. This field is used for reporting purposes, and by Retail Price Management to determine same-sized and different-sized items.	PACKAGE_ UOM	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
DEPOSIT_ ITEM_TYPE	Alpha-numeric	6	N	Deposit item component type. A NULL value in this field indicates that this item is not part of a deposit item relationship. The possible values are: E - Contents A - Container Z - Crate T - Returned item (empty bottle) P - Complex pack (with deposit items) The returned item is flagged only to enable these items to be mapped to a separate general ledger account if required.	DEPOSIT_ITEM _TYPE	VARCHAR2(6)
CONTAINER _ITEM	Alpha-numeric	25	N	Container item number for a contents item. This field is only populated and required if the DEPOSIT_ITEM_ TYPE = E.	CONTAINER_ ITEM	VARCHAR2(25)
DEPOSIT_IN_ PRICE_UOM	Alpha-numeric	6	N	Indicates if the deposit amount is included in the price per UOM calculation for a contents item ticket. This value is only required if the DEPOSIT_ITEM_TYPE = E. Valid values are: I - Include deposit amount E - Exclude deposit amount	DEPOSIT_IN_ PRICE_PER_ UOM	VARCHAR2(6)
RETAIL_ LABEL_TYPE	Alpha-numeric	6	N	Indicates any special label type associated with an item (for example, pre-priced or cents off). This field is used for reporting purposes only. Values for this field are defined by the RTLT code in code detail.	RETAIL_LABEL _TYPE	VARCHAR2(6)
RETAIL_ LABEL_ VALUE	Numeric	20,4	N	Value associated with the retail label type.	RETAIL_LABEL _VALUE	NUMBER(20,4)
PRODUCT_ COMMENTS	Alpha-numeric	2000	N	Comments associated with the product.	COMMENTS	VARCHAR2 (2000)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PERISHABLE_IND	Alpha-numeric	1	N	This field indicates whether the item is perishable or not, Valid values for this field are Y, N. Default = N	PERISHABLE_IND	VARCHAR2(1)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

DC_GROCERY_VARIANT Table

File name: **DC_GROCERY_VARIANT.DAT**

Control file: **DC_GROCERY_VARIANT.CTL**

Staging table: **DC_GROCERY_VARIANT**

Suggested post-loading validation:

- Ensure that ITEM_MASTER.ITEM is unique.
- Ensure that ITEM_MASTER.ITEM_PARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the child less 1.
- Ensure that ITEM_MASTER.ITEM_GRANDPARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the grandchild less 2.
- Ensure that ITEM_MASTER.COST_ZONE_GROUP_ID is a valid COST_ZONE_GROUP..ZONE_GROUP_ID if SYSTEM_OPTIONS.ELC_IND = Y.
- Ensure that ITEM_MASTER.STANDARD_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS is not MISC.
- Ensure that ITEM_MASTER.UOM_CONV_FACTOR is not NULL if UOM_CLASS of ITEM_MASTER.STANDARD_UOM is not QTY.
- Ensure that ITEM_MASTER.PACKAGE_UOM (if not NULL) is a valid UOM_CLASS.UOM.
- Ensure that ITEM_MASTER.RETAIL_LABEL_TYPE (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = RTLT.
- Ensure that ITEM_MASTER.HANDLING_TEMP (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = HTMP.
- Ensure that ITEM_MASTER.HANDLING_SENSITIVITY (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = HSEN.

- Ensure that ITEM_MASTER.ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = UPCT.
- Ensure that ITEM_MASTER.CONTAINER_ITEM is a valid ITEM_MASTER.ITEM if ITEM_MASTER.DEPOSIT_ITEM_TYPE = E.
- Ensure that ITEM_MASTER.FORMAT_ID and ITEM_MASTER.PREFIX are not NULL if ITEM_MASTER.ITEM_NUMBER_TYPE = VPLU.
- Ensure that ITEM_MASTER.FORMAT_ID is a valid VAR_UPC_EAN.FORMAT_ID if ITEM_MASTER.ITEM_NUMBER_TYPE = VPLU.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
VARIANT	Alpha-numeric	25	Y	ID that uniquely identifies the scanning barcode associated with a product.	ITEM	VARCHAR2(25)
VARIANT_NUMBER_TYPE	Alpha-numeric	6	Y	Code specifying what type the variant item is. Valid values for this field are in the code type UPCT in the CODE_HEAD and CODE_DETAIL tables.	ITEM_NUMBER_TYPE	VARCHAR2(6)
VAR_WGT_PLU_FORMAT	Alpha-numeric	1	N	Format ID that corresponds to the item's variable UPC. This value is only used for items with variable UPCs.	FORMAT_ID	VARCHAR2(1)
VAR_WGT_PLU_PREFIX	Integer	2	N	Prefix for variable weight UPCs. The prefix determines the format of the eventual UPC and is used to decode variable weight UPCs that are uploaded from the POS. It is the client's responsibility to download this value to their scale systems.	PREFIX	NUMBER(2)
VARIANT_DESC	Alpha-numeric	250	Y	Description of the variant.	ITEM_DESC	VARCHAR2(250)
VARIANT_SHORT_DESC	Alpha-numeric	120	N	Short description of the variant. Default = First 120 characters of ITEM_DESC	SHORT_DESC	VARCHAR2(120)
VARIANT_DESC_SECONDARY	Alpha-numeric	250	N	Secondary description of the variant.	ITEM_DESC_SECONDARY	VARCHAR2(250)
PRODUCT	Alpha-numeric	25	Y	ID of the product associated with the variant.	ITEM_PARENT	VARCHAR2(25)
PRODUCT_LINE	Alpha-numeric	25	Y	ID of the product line associated with the variant.	ITEM_GRANDPARENT	VARCHAR2(25)
PRIMARY_REF_ITEM_IND	Alpha-numeric	1	N	Default = N Indicates if the variant is the primary variant for the	PRIMARY_REF_ITEM_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
				product.		
VARIANT_COMMENTS	Alpha-numeric	2000	N	Comments associated with the variant.	COMMENTS	VARCHAR2(2000)
AIP_CASE_TYPE	Alpha-numeric	6	N	Only used if AIP is integrated. Determines which case sizes to extract against an item in the AIP interface. Applicable only to non-pack orderable items.	AIP_CASE_TYPE	VARCHAR2(6)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

Load Scripts

DC_PRODUCT_LINE.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_PRODUCT_LINE_FILE – This function call SQLLOADER to load data from input file to DC_PRODUCT_LINE staging table.

LOAD_PRODUCT _FILE – This function call SQLLOADER to load data from input file to DC_PRODUCT staging table.

LOAD_PRODUCT_LINE– This function contains a PL/SQL block that selects from the DC_PRODUCT_LINE and DC_PRODUCT staging tables and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_PRODUCT_LINE and DC_PRODUCT to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	1	NA
TRAN_LEVEL	2	NA
SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
ITEM_AGGREGATE_IND	N	If NULL
DIFF_1_AGGREGATE_IND	N	If NULL
DIFF_2_AGGREGATE_IND	N	If NULL
DIFF_3_AGGREGATE_IND	N	If NULL
DIFF_4_AGGREGATE_IND	N	If NULL
PERISHABLE_IND	N	If NULL

Required file to load: dc_product_line.dat and dc_product.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PRODUCT.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_PRODUCT– This function contains a PL/SQL block that selects from the DC_PRODUCT staging table and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_PRODUCT_LINE and DC_PRODUCT to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	2	NA
TRAN_LEVEL	2	NA

SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
ORDERABLE_IND	Y	NA
SELLABLE_IND	Y	NA
INVENTORY_IND	Y	NA
MERCHANDISE_IND	Y	If NULL
FORECAST_IND	Y	If NULL
ITEM_AGGREGATE_IND	N	NA
DIFF_1_AGGREGATE_IND	N	NA
DIFF_2_AGGREGATE_IND	N	NA
DIFF_3_AGGREGATE_IND	N	NA
DIFF_4_AGGREGATE_IND	N	NA
PRIMARY_REF_ITEM_IND	N	NA
CONST_DIMEN_IND	N	NA
GIFT_WRAP_IND	N	NA
SHIP_ALONE_IND	N	NA
ITEM_XFORM_IND	N	NA
PACK_IND	N	NA
SIMPLE_PACK_IND	N	NA
CATCH_WEIGHT_IND	N	If NULL
CONTAINS_INNER_IND	N	NA
PERISHABLE_IND	N	NA

Required file to load: dc_product_line.dat, dc_product.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_GROCERY_VARIANT.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_GROCERY_VARIANT staging table.

LOAD_GROCERY_VARIANT– This function contains a PL/SQL block that selects from the DC_GROCERY_VARIANT staging table and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_GROCERY_VARIANT and DC_HARDLINES to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_LEVEL	3	NA
TRAN_LEVEL	2	NA
SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
PRIMARY_REF_ITEM_IND	N	If NULL
PERISHABLE_IND	N	If NULL

Required file to load: dc_product_line.dat, dc_product.dat and dc_grocery_variant.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement

DC_DEFAULT_GROCERY.KSH

This ksh script will default VAT_ITEM, UDA_ITEM_LOV and ITEM_CHRG_HEAD/DETAIL for newly created product and product line.

The following functions should be defined in the declaration of the script:

DEFAULT_GROCERY– This function defaults data in the VAT_ITEM, UDA_ITEM_LOV, and ITEM_CHRG_HEAD/DETAIL tables for newly created products and product lines. It includes the following logic:

1. If the VAT indicator is turned on in system_options and default_tax_type is NOT GTAX (i.e. SVAT is used), it uses bulk collect to retrieve into a PL/SQL table the item/department values from the DC_PRODUCT table. It calls the PL/SQL function VAT_SQL.DEFAULT_VAT_ITEM to insert the department VAT defaults into the RMS VAT_ITEM table, by selecting from the vat_deps and vat_code_rates for each item in the DC_PRODUCT table.

2. It also uses bulk collect to retrieve into a PL/SQL table the item/dept/class/subclass values from the DC_PRODUCT and DC_PRODUCT_LINE tables. It calls UDA_SQL.INSERT_DEFAULTS to insert the department UDA defaults

into the RMS uda_item_lov table, by selecting from uda_item_defaults and uda for each item in the DC_PRODUCT and DC_PRODUCT_LINE tables.

3. It calls ITEM_CHARGE_SQL.DEFAULT_CHRGS to insert the department charge defaults into the RMS ITEM_CHRG_HEAD and ITEM_CHRG_DETAIL tables, by selecting from dept_chrg_head and dept_chrg_detail for each item in the DC_PRODUCT and DC_PRODUCT_LINE tables.

Required file to load: dc_product_line.dat, dc_product.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_product_line.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Pack Items

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

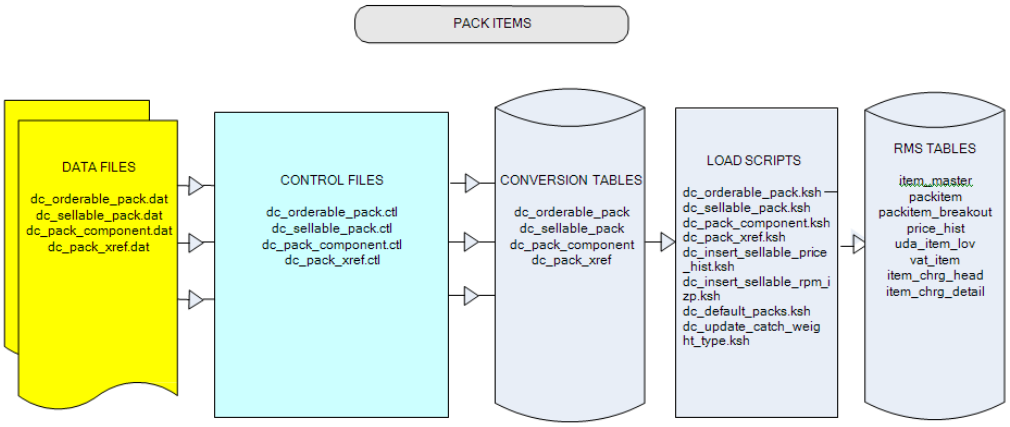
- ITEM_MASTER
- PACKITEM
- PACKITEM_BREAKOUT
- PRICE_HIST
- UDA_ITEM_LOV
- RPM_ITEM_ZONE_PRICE
- VAT_ITEM (only if system_optinos.vat_ind is Y and default_tax_type is not GTAX)
- ITEM_CHRG_HEAD
- ITEM_CHRG_DETAIL

The following programs are included in the Pack Items functional area:

- The following programs are included in the Pack Items functional area:
- Load Scripts:
  - dc_orderable_pack.ksh
  - dc_pack_component.ksh
  - dc_pack_component.ksh
  - dc_pack_xref.ksh
  - dc_insert_sellable_price_hist.ksh
  - dc_insert_sellable_rpm_izp.ksh
  - dc_default_packs.ksh
  - dc_update_catch_weight_type.ksh
- Control Files:
  - dc_orderable_pack.ctl
  - dc_pack_component.ctl
  - dc_pack_component.ctl
  - dc_pack_xref.ctl

The following diagram shows the data flow for the Pack Items functional area:

Data Flow



Data Flow for the Pack Items Functional Area

Prerequisites

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

Staging Table Definition

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_ORDERABLE_PACK Table

File name: **DC_ORDERABLE_PACK.DAT**

This file contains all orderable packs that are either sellable or non-sellable. These packs can be simple packs or complex packs in RMS.

Control file: **DC_ORDERABLE_PACK.CTL**

Staging table: **DC_ORDERABLE_PACK**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL = ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.PACK_IND = Y and ITEM_MASTER.ORDERABLE_IND = Y, and compare to flat file DC_ORDERABLE_PACK.DAT to ensure that all rows are loaded.
- Ensure that ITEM_MASTER.COST_ZONE_GROUP_ID is a valid COST_ZONE_GROUP.ZONE_GROUP_ID if SYSTEM_OPTIONS.ELC_IND = Y and ITEM_MASTER.PACK_IND = Y and ITEM_MASTER.ORDERABLE_IND = Y. Ensure that ITEM_MASTER.DEPT/ITEM_MASTER.CLASS/ ITEM_MASTER.SUBCLASS combination exists in SUBCLASS.
- Ensure that ITEM_MASTER.DIFF_1 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_2 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_3 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.DIFF_4 (if not NULL) is a valid DIFF_IDS.DIFF_ID or DIFF_GROUP_HEAD.DIFF_GROUP_ID.
- Ensure that ITEM_MASTER.PACKAGE_UOM (if not NULL) is a valid UOM_CLASS.UOM.
- Ensure that ITEM_MASTER.RETAIL_LABEL_TYPE (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = RTLT.
- Ensure that ITEM_MASTER.HANDLING_TEMP (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = HTMP.
- Ensure that ITEM_MASTER.HANDLING_SENSITIVITY (if not NULL) is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = HSEN.

FILE FORMAT					**STAGING TABLE DEFINITION**	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PACKID	Alpha-numeric	25	Y	Unique identifier of the pack item.	ITEM	VARCHAR2(25)
DIFF_ 1	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style (for example, stone-washed). Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_1	VARCHAR2(10)
DIFF_ 2	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_2	VARCHAR2(10)
DIFF_3	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_3	VARCHAR2(10)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
DIFF_4	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are found in the DIFF_GROUP and DIFF_IDS tables.	DIFF_4	VARCHAR2(10)
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 that is a member of merchandise hierarchy level 6. Valid values are in the DEPT field in the DEPS table in RMS.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifier of the merchandise hierarchy level 5 that is a member of merchandise hierarchy level 4. Valid values are in the CLASS field in the CLASS table in RMS.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Identifier of the merchandise hierarchy level 6 that is a member of merchandise hierarchy level 5. Valid values are in the SUBCLASS field in the SUBCLASS table in RMS.	SUBCLASS	NUMBER(4)
PACK_DESCRIPTION	Alpha-numeric	250	Y	Description of the pack item.	ITEM_DESC	VARCHAR2(250)
PACK_SHORT_DESC	Alpha-numeric	120	N	Short description of the pack item. Default = First 120 characters of PACK_DESC	SHORT_DESC	VARCHAR2(120)
PACK_SECONDARY_DESC	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
COST_ZONE_GROUP_ID	Integer	4	N	NULL if PACK_TYPE = Buyer; otherwise NOT NULL. Cost zone group associated with the item. This field is only required when ELC_IND (landed cost indicator) is set to Y in the SYSTEM_OPTIONS table.	COST_ZONE_GROUP_ID	NUMBER(4)
PACKAGE_SIZE	Numeric	12,4	N	Size of the product printed on any packaging (for example, 24 ounces). This field is used for reporting purposes, as well as by Retail Price Management to determine same-sized and different-sized items.	PACKAGE_SIZE	NUMBER(12,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PACKAGE_UOM	Alpha-numeric	4	N	Unit of measure associated with the package size. This field is used for reporting purposes, and by Retail Price Management to determine same-sized and different-sized items.	PACKAGE_UOM	VARCHAR2(4)
STORE_ORD_MULT	Alpha-numeric	1	N	Unit type in which products shipped from the warehouses to the stores must be specified. Valid values are: C - Cases I - Inner E - Eaches Default = E	STORE_ORD_MULT	VARCHAR2(1)
MFG_REC_RETAIL	Numeric	20,4	N	Manufacturer's recommended retail price for the item. Used for information only. Must be in the primary currency. NULL if SELLABLE_IND = N	MFG_REC_RETAIL	NUMBER(20,4)
RETAIL_LABEL_TYPE	Alpha-numeric	6	N	Any special label type associated with an item (for example, pre-priced or cents off). This field is used for reporting purposes only. Values for this field are defined by the RTLT code on code detail. NULL if SELLABLE_IND = N	RETAIL_LABEL_TYPE	VARCHAR2(6)
RETAIL_LABEL_VALUE	Numeric	20,4	N	The value associated with the retail label type. NULL if SELLABLE_IND = N	RETAIL_LABEL_VALUE	NUMBER(20,4)
HANDLING_TEMP	Alpha-numeric	6	N	Temperature information associated with the item. Valid values for this field are in the code type HTMP in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_TEMP	VARCHAR2(6)
HANDLING_SENSITIVITY	Alpha-numeric	6	N	Sensitivity information associated with the item. Valid values for this field are in the code type HSEN in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_SENSITIVITY	VARCHAR2(6)
CATCH_WEIGHT_IND	Alpha-numeric	1	Y	Indicates whether the item should be weighed when it arrives at a location. Valid values for this field are Y and N.	CATCH_WEIGHT_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
SIMPLE_PACK_IND	Alpha-numeric	1	Y	Indicates whether the pack item contains all the same items within (simple) or the pack item has different items (complex). Valid values are Y or N.	SIMPLE_PACK_IND	VARCHAR2(1)
SELLABLE_IND	Alpha-numeric	1	Y	Indicates whether the pack item is sellable to a customer.	SELLABLE_IND	VARCHAR2(1)
PACK_TYPE	Alpha-numeric	1	N	Indicates whether the pack item is a vendor pack or a buyer pack. Valid values are: B - Buyer V - Vendor Required (V or B) for a complex pack: V for a simple pack. B for a buyer pack that is either "Assembled as a pack after receipt and ordered as individual items," or "Vendor pack," where the pack is ordered.	PACK_TYPE	VARCHAR2(1)
ORDER_AS_TYPE	Alpha-numeric	1	N	Indicates whether a pack item is receivable at the component level or at the pack level (for a buyer pack only). This field is required if the pack item is an orderable buyer pack. This field must be NULL if the pack is sellable only or a vendor pack. Valid values are: E - Eaches (component level) P - Pack (buyer pack only) Identifies whether a buyer pack should be ordered as the components of the pack (E), or the pack item should be ordered (P). For example, pack A contains 6 of item B and 6 of item C. If this field is P, the order would be placed for item A. If this field is E, when ordering 1 unit of A, the supplier would actually receive the order for 6 B items and 6 C items.	ORDER_AS_TYPE	VARCHAR2(1)
PACK_COMMENTS	Alpha-numeric	2000	N	Any comments associated with the pack item	COMMENTS	VARCHAR2 (2000)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
CATCH_WEIGHT_ORDER_TYPE	Alpha-numeric	6	N	How catch weight items are ordered. Valid values are in the CODE_DETAIL table with a code type ORDT. NOT NULL if CATCH_WEIGHT_IND = Y	ORDER_TYPE	VARCHAR2(6)
CATCH_WEIGHT_SALE_TYPE	Alpha-numeric	6	N	Method for selling catch weight items in store locations. Valid values are in the CODE_DETAIL table with a code type STYP. NULL if non-sellable.	SALE_TYPE	VARCHAR2(6)
NOTIONAL_PACK_IND	Alpha-numeric	1	N	This is to indicate that the pack item should post the transaction at component level in SIM. Valid values for this field are Y, N. Default value is N (if NULL in External Table).	NOTIONAL_PACK_IND	VARCHAR2(1)
SOH_INQUIRY_AT_PACK_IND	Alpha-numeric	1	N	This indicates to show the stock on hand at pack level in downstream applications when it is called in POS from SIM. Valid values for this field are Y, N. If field value is Y then the notional_pack_ind also should be Y. Default value is N (if NULL in External Table).	SOH_INQUIRY_AT_PACK_IND	VARCHAR2(1)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

DC_SELLABLE_PACK Table

File name: **DC_SELLABLE_PACK.DAT**

This file contains all sellable packs that are non-orderable. These packs can only be complex packs in RMS.

Control file: **DC_SELLABLE_PACK.CTL**

Staging table: **DC_SELLABLE_PACK**

Suggested post-loading validation:

- Capture counts from ITEM_MASTER where ITEM_MASTER.ITEM_LEVEL = ITEM_MASTER.TRAN_LEVEL and ITEM_MASTER.PACK_IND = Y and ITEM_MASTER.ORDERABLE_IND = N, and compare to flat file DC_SELLABLE_PACK.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PACKID	Alpha-numeric	25	Y	Unique identifier of the pack item	ITEM	VARCHAR2(25)
DIFF_ 1	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style (for example, stone-washed). Valid values are in the DIFF_GROUP and DIFF_IDS tables.	DIFF_1	VARCHAR2(10)
DIFF_ 2	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are in the DIFF_GROUP and DIFF_IDS tables.	DIFF_2	VARCHAR2(10)
DIFF_3	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are in the DIFF_GROUP and DIFF_IDS tables.	DIFF_3	VARCHAR2(10)
DIFF_4	Alpha-numeric	10	N	ID of the differentiator that differentiates the SKU from its Style. Valid values are in the DIFF_GROUP and DIFF_IDS tables.	DIFF_4	VARCHAR2(10)
MERCH_HIER_4	Integer	4	Y	Identifier of the merchandise hierarchy level 4 that is a member of merchandise hierarchy level 6. Valid values are in the DEPT field in the DEPS table in RMS.	DEPT	NUMBER(4)
MERCH_HIER_5	Integer	4	Y	Identifier of the merchandise hierarchy level 5 that is a member of merchandise hierarchy level 4. Valid values are in the CLASS field in the CLASS table in RMS.	CLASS	NUMBER(4)
MERCH_HIER_6	Integer	4	Y	Identifier of the merchandise hierarchy level 6 that is a member of merchandise hierarchy level 5. Valid values are in the SUBCLASS field in the SUBCLASS table in RMS.	SUBCLASS	NUMBER(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PACK_DESCRIPTION	Alpha-numeric	250	Y	Description of the pack item.	ITEM_DESC	VARCHAR2(250)
PACK_SHORT_DESC	Alpha-numeric	120	N	Short description of the pack item. Default = First 120 char of PACK_DESC.	SHORT_DESC	VARCHAR2(120)
PACK_SECONDARY_DESC	Alpha-numeric	250	N	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
PACKAGE_SIZE	Numeric	12,4	N	Size of the product printed on any packaging (for example, 24 ounces). This field is used for reporting purposes, as well as by RPM to determine same-sized and different-sized items.	PACKAGE_SIZE	NUMBER(12,4)
PACKAGE_UOM	Alpha-numeric	4	N	Unit of measure associated with the package size. This field is used for reporting purposes, and by RPM to determine same sized and different sized items.	PACKAGE_UOM	VARCHAR2(4)
MFG_REC_RETAIL	Numeric	20,4	N	Manufacturer's recommended retail price for the item. Used for information only. Needs to be in the primary currency. NULL if SELLABLE_IND = N	MFG_REC_RETAIL	NUMBER(20,4)
RETAIL_LABEL_TYPE	Alpha-numeric	6	N	Any special label type associated with an item (for example, pre-priced or cents off). This field is used for reporting purposes only. Values for this field are defined by the RTLT code on code detail. NULL if SELLABLE_IND = N	RETAIL_LABEL_TYPE	VARCHAR2(6)
RETAIL_LABEL_VALUE	Numeric	20,4	N	Value associated with the retail label type. NULL if SELLABLE_IND = N	RETAIL_LABEL_VALUE	NUMBER(20,4)
HANDLING_TEMP	Alpha-numeric	6	N	Temperature information associated with the item. Valid values for this field are in the code type HTMP in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_TEMP	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
HANDLING_SENSITIVITY	Alpha-numeric	6	N	Sensitivity information associated with the item. Valid values for this field are in the code type HSEN in the CODE_HEAD and CODE_DETAIL tables.	HANDLING_SENSITIVITY	VARCHAR2(6)
UNIT_RETAIL	Numeric	20,4	Y	Item's current unit retail in the system's primary currency.	UNIT_RETAIL	NUMBER(20,4)
PACK_COMMENTS	Alpha-numeric	2000	N	Comments related to the pack item.	COMMENTS	VARCHAR2(2000)
PERISHABLE_IND	Alpha-numeric	1	N	A grocery item attribute used to indicate whether an item is perishable or not.	PERISHABLE_IND	VARCHAR2(1)
NOTIONAL_PACK_IND	Alpha-numeric	1	N	This is to indicate that the pack item should post the transaction at pack level in SIM. If this indicator is checked in RMS, SIM will track pack item at the pack level. If the indicator is not checked in RMS, SIM will store inventory at the component level.	NOTIONAL_PACK_IND	VARCHAR2(1)
SOH_INQUIRY_AT_PACK_IND	Alpha-numeric	1	N	This indicates to show the stock on hand at pack level in downstream applications when it is called in POS from SIM.	SOH_INQUIRY_AT_PACK_IND	VARCHAR2(1)
PRODUCT_CLASSIFICATION	Alpha-numeric	6	N	This Column contains item combinability codes (with code type PCLA) which provide a way to define which items can be combined (packed or boxed) together and communicate the same to WMS.	PRODUCT_CLASSIFICATION	VARCHAR2(6)
BRAND_NAME	Alpha-numeric	30	N	This is used to associate a brand to an item.	BRAND_NAME	VARCHAR2(30)

DC_PACK_COMPONENT Table

File name: **DC_PACK_COMPONENT.DAT**

Control file: **DC_PACK_COMPONENT.CTL**

Staging table: **DC_PACK_COMPONENT**

Suggested post-loading validation:

- Capture counts from PACK_ITEM and compare to flat file DC_PACK_COMPONENT.DAT to ensure that all rows are loaded.

- Ensure that PACK_ITEM.PACK_NO is a valid ITEM_MASTER.ITEM where ITEM_MASTER.PACK_IND = Y.
- Ensure that PACK_ITEM.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.TRAN_LEVEL = ITEM_MASTER.ITEM_LEVEL.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PACK_ID	Alpha-numeric	25	Y	ID of the pack item.	PACK_NO	VARCHAR2(25)
ITEM	Alpha-numeric	25	Y	ID of the item contained in the pack.	ITEM	VARCHAR2(25)
ITEM_QTY	Alpha-numeric	12,4	Y	Quantity of the item within the pack.	PACK_ITEM_QTY	NUMBER(12,4)

Note: If any records are in the BAD or DISCARD file, the RMS table must be truncated the entire file must be rerun. No new records within a sequence group can be added to the RMS table through the scripts.

DC_PACK_XREF Table

File name: **DC_PACK_XREF.DAT**

Control file: **DC_PACK_XREF.CTL**

Staging table: **DC_PACK_XREF**

Suggested post-loading validation:

- Ensure that ITEM_MASTER.ITEM is unique.
- Ensure that ITEM_MASTER.ITEM_PARENT (if not NULL) is a valid ITEM_MASTER.ITEM with ITEM_MASTER.ITEM_LEVEL = item level of the child less 1.
- Ensure that ITEM_MASTER.ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = UPCT.
- Ensure that ITEM_MASTER.FORMAT_ID and ITEM_MASTER.PREFIX are not NULL if ITEM_MASTER.ITEM_NUMBER_TYPE = VPLU.
- Ensure that ITEM_MASTER.FORMAT_ID is a valid VAR_UPC_EAN.FORMAT_ID if ITEM_MASTER.ITEM_NUMBER_TYPE = VPLU.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
XREF_PACK	Alpha-numeric	25	Y	ID that uniquely identifies the scanning barcode associated with a product.	ITEM	VARCHAR2(25)
XREF_DESC	Alpha-numeric	250	Y	Description of the item.	ITEM_DESC	VARCHAR2(250)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
XREF_SHORT_DESC	Alpha-numeric	120	N	Default = 120 char of XREF_DESC.	SHORT_DESC	VARCHAR2(120)
XREF_SECOND_DESC	Alpha-numeric	250	Y	Secondary description of the SKU for Yomi requirement.	ITEM_DESC_SECONDARY	VARCHAR2(250)
PACK_ID	Alpha-numeric	25	Y	Pack item associated with the xref item.	ITEM_PARENT	VARCHAR2(25)
XREF_COMMENTS	Alpha-numeric	2000	N	Comments attached to the xref item.	COMMENTS	VARCHAR2(2000)
PRIMARY_REF_ITEM_IND	Alpha-numeric	1	N	There can be many xref items for a pack item; this indicates whether this is the primary xref item. Default = N	PRIMARY_REF_IND	VARCHAR2(1)
ITEM_NUMBER_TYPE	Alpha-numeric	6	Y	Code specifying what type the XREF_PACK is. Valid values for this field are in the code type UPCT in the CODE_HEAD and CODE_DETAIL tables.	ITEM_NUMBER_TYPE	VARCHAR2(6)
VAR_WGT_PLU_FORMAT	Alpha-numeric	6	N	Format ID that corresponds to the item's variable UPC. This value is only used for items with variable UPCs.	FORMAT_ID	VARCHAR2(1)
VAR_WGT_PLU_PREFIX	Integer	2	N	Prefix for variable weight UPCs. The prefix determines the format of the eventual UPC and is used to decode variable weight UPCs that are uploaded from the POS.	PREFIX	NUMBER(2)

Load Scripts

DC_ORDERABLE_PACK.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ORDERABLE_PACK staging table.

LOAD_ORDERABLE_PACK– This function contains a PL/SQL block that selects from the DC_ORDERABLE_PACK staging table and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ORDERABLE_PACK to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	1	NA
TRAN_LEVEL	1	NA
SHORT_DESC	substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	Sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	Sysdate	NA
MFG_REC_RETAIL	NA	Ensure that sellable_ind = Y, otherwise NULL
COST_ZONE_GROUP_ID		Ensure that orderable_ind = Y and pack_type != B, otherwise null
STANDARD_UOM	EA	NA
STORE_ORD_MULT	NA	If NULL, E
ORDERABLE_IND	Y	NA
INVENTORY_IND	Y	NA
PACK_TYPE	NA	Ensure V if simple pack
ORDER_AS_TYPE	NA	Ensure pack_type = B and
ITEM_AGGREGATE_IND	N	NA
DIFF_1_AGGREGATE_IND	N	NA
DIFF_2_AGGREGATE_IND	N	NA
DIFF_3_AGGREGATE_IND	N	NA
DIFF_4_AGGREGATE_IND	N	NA
PRIMARY_REF_ITEM_IND	N	NA
CONST_DIMEN_IND	N	NA
GIFT_WRAP_IND	N	NA
SHIP_ALONE_IND	N	NA

Column Name (RMS Table)	Default Value	Comments
ITEM_XFORM_IND	N	NA

Required file to load: dc_orderable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_SELLABLE_PACK.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_SELLABLE_PACK staging table.

LOAD_VAT_DEPS– This function contains a PL/SQL block that selects from the DC_SELLABLE_PACK staging table and inserts the data to the RMS ITEM_MASTER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_SELLABLE_PACK to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	NA
ITEM_LEVEL	1	NA
TRAN_LEVEL	1	NA

Required file to load: dc_sellable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PACK_COMPONENT.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PACK_COMPONENT staging table.

LOAD_PACK_COMPONENT– This function contains a PL/SQL block that selects from the DC_PACK_COMPONENT staging tables and inserts the data to the RMS PACKITEM and PACKITEM_BREAKOUT tables.

Because inner packs are not supported as part of the data conversion toolset, the RMS tables PACKITEM and PACKITEM_BREAKOUT have the same data after loading.

> **Note**: If the loading of DC_PACK_COMPONENT results in any bad data, the PACKITEM and PACKITEM_BREAKOUT tables should be truncated. The bad data should be fixed in the original data file and loaded again. This ensures that the correct seq_no is generated and inserted into RMS tables.

It is assumed that all component items in the DC_PACK_COMPONENT table have been loaded as approved items with data in the ITEM_MASTER and ITEM_SUPP_COUNTRY tables, and that the components for each of the packs in DC_SELLABLE_PACK and DC_ORDERABLE_PACK are included in this table. If not, the data will be inconsistent.

Most of the columns from the staging table defined above directly map to the RMS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_PACK_COMPONENT to PACKITEM and PACKITEM_BREAKOUTColumn Defaults

Column Name (RMS Table)	Default Value	Comments
SEQ_NO		Seq no + 1 for each unique item use analytic function row number ()
CREATE_DATETIME	Sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	Sysdate	NA

Required file to load: dc_pack_component.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PACK_XREF.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PACK_XREF staging table.

LOAD_PACK_XREF– This function contains a PL/SQL block that selects from the DC_PACK_XREF and DC_PACK staging tables and inserts the data to the RMS ITEM_MASTER table. Most of the columns from the staging table defined above directly map to the RMS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_PACK_XREF and DC_PACK to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_LEVEL	2	NA
TRAN_LEVEL	1	NA
SHORT_DESC	SUBSTR 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	NA
STATUS	A	NA
CREATE_DATETIME	sysdate	NA
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	sysdate	NA
PERISHABLE_IND	N	NA

Required file to load: dc_pack_xref.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_SELLABLE_PRICE_HIST.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_SELLABLE_PRICE_HIST– This function inserts the 0 tran_type, 0 reason, 0 location record into the RMS PRICE_HIST table only for sellable non-orderable packs. (All other items have this record inserted with the ITEM_SUPPLIER load script). It retrieves the items from the DC_SELLABLE_PACK table. For each item, it calls the PACKITEM_ADD_SQL.BUILD_COMP_COST_RETAIL function to retrieve the UNIT_COST and UNIT_RETAIL in the primary currency. It uses these values for the 0 record in PRICE_HIST for the UNIT_COST and UNIT_RETAIL.

The pack's UNIT_COST and UNIT_RETAIL are determined from the pack components. It is assumed that all component items in the DC_PACK_COMPONENT table have been loaded as approved items with data in the ITEM_MASTER and ITEM_SUPP_COUNTRY tables, and that the components for each of the packs in DC_SELLABLE_PACK are included in this table. If not the data will be inconsistent.

The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_SELLABLE_PACK to PRICE_HIST Column Defaults

Column Name (RMS Table)	Default Value	Comments
ACTION_DATE	VDATE	NA
TRAN_TYPE	0	NA
LOC	0	NA
REASON	0	NA
SELLING_UNIT_RETAIL	Unit_retail	NA
SELLING_UOM	EA	NA

Required file to load: dc_sellable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_SELLABLE_RPM_IZP.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_SELLABLE_RPM_IZP– This function selects from the DC_SELLABLE_PACK staging table and joins with RPM_MERCH_RETAIL_DEF to insert data to the RPM_ITEM_ZONE_PRICE table. This function retrieves the regular zone group ID for the department of the items in the DC_SELLABLE_PACK table, and joins with the RPM_MERCH_RETAIL_DEF_EXPL view to get the regular RPM GROUP_ZONE_ID for the item's department/class/subclass, performs a bulk collect of this data and loops through the results to insert into the RPM_ITEM_ZONE_PRICE table. For the insert/select, it joins DC_SELLABLE_PACK for each item and the RPM_ZONE for the department's ZONE_GROUP_ID.

The function retrieves the primary currency from SYSTEM_OPTIONS table. If the zone currency and the primary currency are different, UNIT_RETAIL is converted to the zone currency. The following table indicates the data retrieved for value insert.

DC_SELLABLE_PACK to RPM_ITEM_ZONE_PRICE Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_ZONE_PRICE_ID	Use sequence	NA
ITEM	dc_sellable_pack.item	NA
ZONE_ID	Rpm_zone.zone_id	For the department zone_group_id
STANDARD_RETAIL	dc_sellable_pack.unit_retail	NA
STANDARD_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
STANDARD_UOM	EA	NA
SELLING_RETAIL	dc_sellable_pack.unit_retail	NA

Column Name (RMS Table)	Default Value	Comments
SELLING_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
SELLING_UOM	EA	NA
MULTI_UNIT_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id

Required file to load: dc_sellable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_DEFAULT_PACKS.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

DEFAULT_PACKS– This function inserts item defaults from the merchandise hierarchy specifications for UDAs, VAT (if the default_tax_type != GTAX) and for ITEM CHARGES (non-buyer packs only).

This retrieves the ITEM, DEPT, CLASS and SUBCLASS values from DC_ORDERBLE_PACK and DC_SELLABLE_PACK. It calls UDA_SQL.INSERT_DEFAULTS for both sellable and orderable packs. If the default_tax_type != GTAX (SVAT is used), then it calls VAT_SQL.DEFAULT_VAT_ITEM for both sellable and orderable packs.

This also retrieves SKU and dept information for non-buyer packs. Calls ITEM_CHARGE_SQL.DEFAULT_CHARGES.

Required file to load: dc_orderable_pack.dat, dc_sellable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UPDAE_CATCH_WEIGHT_TYPE.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

UPDATE_CATCH_WEIGHT_TYPE– This function updates the ITEM_MASTER table for those records that have been inserted by the DC_ORDERABLE_PACK.KSH function.

The update to the ITEM_MASTER table takes place in the CATCH_WEIGHT_TYPE column when a catch weight simple pack is created in RMS. The updated value is 2 or 4 for simple pack catch weight items (or NULL for other items), depending on the sale type and STANDARD_UOM of the component item at the time of approval.

Updates occur for items inserted in Approved status, and where CATCH_WEIGHT_IND=Y, SIMPLE_PACK_IND=Y, PACK_TYPE=V, and ORDER_TYPE=V.

Required file to load: dc_orderable_pack.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_orderable_pack.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Item Supplier Overview

This section describes data conversion for the following tables, listed in the order in which they must be loaded:

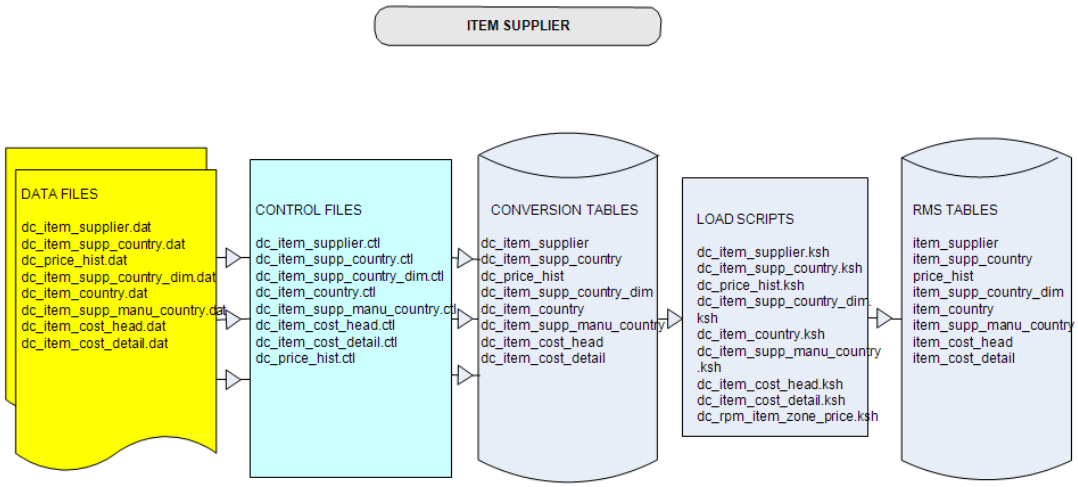
- ITEM_COUNTRY
- ITEM_SUPPLIER
- ITEM_SUPP_COUNTRY
- ITEM_SUPP_MANU_COUNTRY
- ITEM_COST_HEAD
- ITEM_COST_DETAIL
- ITEM_SUPP_COUNTRY_DIM
- RPM_ITEM_ZONE_PRICE
- PRICE_HIST

The following programs are included in this functional area.

- Load Scripts:
  - dc_item_country.ksh
  - dc_item_supplier.ksh
  - dc_item_supp_country.ksh
  - dc_item_supp_manu_country.ksh
  - dc_item_supp_country_dim.ksh
  - dc_item_cost_head.ksh
  - dc_item_cost_detail.ksh
  - dc_price_hist.ksh
  - dc_ rpm_item_zone_price.ksh
- Control Files:
  - dc_item_country.ctl
  - dc_item_supplier.ctl
  - dc_item_supp_country.ctl
  - dc_item_supp_manu_country.ctl
  - dc_item_supp_country_dim.ctl
  - dc_item_cost_head.ctl
  - dc_item_cost_detail.ctl
  - dc_price_hist.ctl

Data Flow

The following diagram shows the data flow for the Item Supplier functional area:



Data Flow for the Item Supplier Functional Area

Prerequisites

Before you begin using the data conversion toolset for Item Supplier, you must complete data conversion for the following:

- Fashion Items
- Hardlines
- Grocery Items
- Pack Items

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

STAGING TABLE DEFINITION

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_ITEM_SUPPLIER Table

File name: **DC_ITEM_SUPPLIER.DAT**

Control file: **DC_ITEM_SUPPLIER.CTL**

Staging table: **DC_ITEM_SUPPLIER**

> **Note:** DC_ITEM_SUPPLIER must have a row/record for every item level, including below-transaction level (reference items).

Suggested post-loading validation:

- Capture counts from ITEM_SUPPLIER and compare to flat file DC_ITEM_SUPPLIER.DAT to ensure that all rows are loaded.
- Ensure that ITEM_SUPPLIER.ITEM is a valid ITEM_MASTER.ITEM.
- Ensure that ITEM_SUPPLIER.SUPPLIER is a valid SUPS.SUPPLIER.
- Ensure that ITEM_SUPPLIER.PALLET_NAME is a valid CODE_DETAIL.CODE where CODE_TYPE = PALN.
- Ensure that ITEM_SUPPLIER.PALLET_NAME is a valid CODE_DETAIL.CODE where CODE_TYPE = PALN.
- Ensure that ITEM_SUPPLIER.PALLET_NAME is a valid CODE_DETAIL.CODE where CODE_TYPE = PALN.
- Ensure that ITEM_SUPPLIER.CASE_NAME is a valid CODE_DETAIL.CODE where CODE_TYPE = CASN.
- Ensure that ITEM_SUPPLIER.INNER_NAME is a valid CODE_DETAIL.CODE where CODE_TYPE = INRN.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
SKU	Alpha-numeric	25	Y	ID of the stock keeping unit.	ITEM	VARCHAR2(25)
SUPPLIER	Integer	10	Y	ID of the supplier that supplies the SKU.	SUPPLIER	NUMBER(10)
PALLET_NAME	Alpha-numeric	6	N	Code referencing the name used to refer to the pallet. Valid codes are defined in the PALN code type. Examples are flat, pallet. Default from System Options.	PALLET_NAME	VARCHAR2(6)
CASE_NAME	Alpha-numeric	6	N	Code referencing the name used to refer to the case. Valid codes are defined in the CASN code type. Examples are pack, box, and bag. Default from System Options.	CASE_NAME	VARCHAR2(6)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
INNER_NAME	Alpha-numeric	6	N	Code referencing the name used to refer to the inner. Valid codes are defined in the INRN code type. Examples are sub-case, sub-pack. Default from System Options.	INNER_NAME	VARCHAR2(6)
DIRECT_SHIP_IND	Alpha-numeric	1	N	Indicates whether any item associated with this supplier is eligible for a direct shipment from the supplier to the customer. Default = N	DIRECT_SHIP_IND	VARCHAR2(1)
VPN	Alpha-numeric	30	N	Vendor product number associated with the SKU.	VPN	VARCHAR2(30)
CONCESSION_RATE	Numeric	12,4	N	Margin that the supplier receives on sale of the item. If the SKU is a concession item, this field is required.	CONCESSION_RATE	NUMBER(12,4)
SUPP_LABEL	Alpha-numeric	15	N	Supplier label. This field can only have a value if the item is a style.	SUPP_LABEL	VARCHAR2(15)
CONSIGNMENT_RATE	Numeric	12,4	N	Consignment rate for this item for the supplier. If the item is a consignment item, this field is required.	CONSIGNMENT_RATE	NUMBER(12,4)
PRIMARY_SUPP_IND	Alpha-numeric	1	N	Indicates whether this supplier is the primary supplier for the item. Each item can have only one primary supplier. Valid values are Y and N. Lowest Supplier ID = Y, otherwise default = N. **Note:** This column must either be populated for all records or NULL for all records.	PRIMARY_SUPP_IND	VARCHAR2(1)

Note: If a record is in the BAD or DISCARD file and the PRIMARY_SUPP_IND is NULL in the file, then the record must be populated with N to be loaded, or the RMS table must be truncated and the entire file must be rerun.

DC_ITEM_SUPP_COUNTRY Table

File name: **DC_ITEM_SUPP_COUNTRY.DAT**

Control file: Control File: **DC_ITEM_SUPP_COUNTRY.CTL**

Staging table: **DC_ITEM_SUPP_COUNTRY**

> **Note:** The DC_ITEM_SUPP_COUNTRY table must have rows/records for item levels that are transaction level or above. There should not be any data for below-transaction-level items.

Suggested post-loading validation:

- Capture counts from ITEM_SUPP_COUNTRY and will create the DC_ITEM_SUPP_COUNTRY oracle external table.DAT to ensure that all rows are loaded.
- Ensure that ITEM_SUPPLIER.ITEM is a valid ITEM_MASTER.ITEM.
- Ensure that ITEM_SUPPLIER.SUPPLIER is a valid SUPS.SUPPLIER.
- Ensure that ITEM_SUPP_COUNTRY.ITEM/ITEM_SUPP_COUNTRY.SUPPLIER combination exists on ITEM_SUPPLIER.
- Ensure that ITEM_SUPP_COUNTRY.ORIGIN_COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.
- Ensure that ITEM_SUPP_COUNTRY.PACKING_METHOD is a valid CODE_DETAIL.CODE where CODE_TYPE = PKMT

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
SKU	Alpha-numeric	25	Y	ID of the stock Keeping Unit	ITEM	VARCHAR2(25)
SUPPLIER	Integer	10	Y	ID of the supplier that supplies the SKU	SUPPLIER	NUMBER(10)
ORIGIN_COUNTRY_ID	Alpha-numeric	3	Y	ID of the country where the item is sourced i.e. the country where the supplier is based	ORIGIN_COUNTRY_ID	VARCHAR2(3)
UNIT_COST	Numeric	20,4	Y	This field contains the current corporate unit cost for the SKU from the supplier/origin country. This field is stored in the supplier's currency.	UNIT_COST	NUMBER(20,4)
SUPP_PACK_SIZE	Numeric	12,4	Y	This field contains the quantity that orders must be placed in multiples of for the supplier for the item.	SUPP_PACK_SIZE	NUMBER(12,4)
INNER_PACK_SIZE	Numeric	12,4	Y	This field contains the break pack size for this item from the supplier.	INNER_PACK_SIZE	NUMBER(12,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ROUND_LVL	Alpha-numeric	6	N	This column will be used to determine how order quantities will be rounded to Case, Layer and Pallet. Valid values are: ▪ C ▪ L ▪ P ▪ Cl ▪ LP ▪ CLP Default from System Options.	ROUND_LVL	VARCHAR2(6)
ROUND_TO_INNER_PCT	Numeric	12,4	N	This column will hold the Inner Rounding Threshold value. During rounding, this value is used to determine whether to round partial Inner quantities up or down. If the Inner-fraction in question is less than the Threshold proportion, it is rounded down; if not, it is rounded up. For instance, with an Inner size of 10 and a Threshold of 80%, Inner quantities such as 18, 29 and 8 would be rounded up to 20, 30 and 10 respectively, while quantities of 12, 27 and 35 would be rounded down to 10, 20 and 30 respectively. Quantities are never rounded down to zero; a quantity of 7, in the example above, would be rounded up to 10. This column will be maintained simply for the purpose of defaulting to the Item/Supplier/Country/Location level. Default from System Options.	ROUND_TO_INNER_PCT	NUMBER(12,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ROUND_TO_CASE_PCT	Numeric	12,4	N	This column will hold the Case Rounding Threshold value. During rounding, this value is used to determine whether to round partial Case quantities up or down. If the Case-fraction in question is less than the Threshold proportion, it is rounded down; if not, it is rounded up. For instance, with an Case size of 10 and a Threshold of 80%, Case quantities such as 18, 29 and 8 would be rounded up to 20, 30 and 10 respectively, while quantities of 12, 27 and 35 would be rounded down to 10, 20 and 30 respectively. Quantities are never rounded down to zero; a quantity of 7, in the example above, would be rounded up to 10. This column will be maintained simply for the purpose of defaulting to the Item/Supplier/Country/ Location level. Default from System Options.	ROUND_TO_CASE_PCT	NUMBER(12,4)
ROUND_TO_LAYER_PCT	Numeric	12,4	N	This column will hold the Layer Rounding Threshold value. During rounding, this value is used to determine whether to round partial Layer quantities up or down. If the Layer-fraction in question is less than the Threshold proportion, it is rounded down; if not, it is rounded up. Default from System Options.	ROUND_TO_LAYER_PCT	NUMBER(12,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ROUND_TO_PALLET_PCT	Numeric	12,4	N	This column will hold the Pallet Rounding Threshold value. During rounding, this value is used to determine whether to round partial Pallet quantities up or down. If the Pallet - fraction in question is less than the Threshold proportion, it is rounded down; if not, it is rounded up. For instance, with an Pallet size of 10 and a Threshold of 80%, Pallet quantities such as 18, 29 and 8 would be rounded up to 20, 30 and 10 respectively, while quantities of 12, 27 and 35 would be rounded down to 10, 20 and 30 respectively. Quantities are never rounded down to zero; a quantity of 7, in the example above, would be rounded up to 10. This column will be maintained simply for the purpose of defaulting to the Item/Supplier/Country/ Location level. Default from System Options.	ROUND_TO_PALLET_PCT	NUMBER(12,4)
MIN_ORDER_QTY	Numeric	12,4	N	This field contains the minimum allowable order quantity for the item from the supplier. This parameter is used for order quantity validations.	MIN_ORDER_QTY	NUMBER(12,4)
MAX_ORDER_QTY	Numeric	12,4	N	This field contains the maximum allowable order quantity for the item from the supplier. This parameter is used for order quantity validations.	MIN_ORDER_QTY	NUMBER(12,4)
PRIMARY_COUNTRY_IND	Alpha-numeric	1	N	This field indicates whether this country is the primary country for the item/supplier. Each item/supplier combination must have one and only one primary country. Valid values are Y or N. First Alpha Country ID = Y otherwise Default = N This column must either be entered for All records, or Null for all records.	PRIMARY_COUNTRY_IND	VARCHAR2(1)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TI	Numeric	12,4	Y	Number of shipping units (cases) that make up one tier of a pallet. Multiply TI x HI to get total number of units (cases) for a pallet.	TI	NUMBER(12,4)
HI	Numeric	12,4	Y	Number of tiers that make up a complete pallet (height). Multiply TI x HI to get total number of units (cases) for a pallet.	HI	NUMBER(12,4)
COST_UOM	Alpha-numeric	4	N	A cost UOM is held to allow cost to be managed in a separate UOM to the standard UOM Default to standard UOM (item_master)	COST_UOM	VARCHAR2(4)
LEAD_TIME	Integer	4	N	This field contains the number of days that will elapse between the date an order is written and the delivery to the store or warehouse from the supplier. Default from SUPS	LEAD_TIME	NUMBER(4)
PACKING_METHOD	Alpha-numeric	6	N	This field indicates whether the packing method of the item in the container is Flat or Hanging. Values for this field are store in the PKMT code. Default from System Options	PACKING_METHOD	VARCHAR2(6)
DEFAULT_UOP	Alpha-numeric	6	N	Contains the default unit of purchase for the item/supplier/country. Valid values include: Standard Units of Measure C for Case P for Pallet Default = C	DEFAULT_UOP	VARCHAR2(6)
NEGOTIATED_ITEM_COST	Numeric	20,4	N	This will hold the supplier negotiated item cost for the primary delivery country of the item. Once a location is associated with the item, the primary locations negotiated item cost will be stored in this field.	NEGOTIATED_ITEM_COST	NUMBER(20,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
EXTENDED_BASE_COST	Numeric	20,4	N	This will hold the extended base cost for the primary delivery country of the item. Once a location is associated with the item, the primary locations extended base cost will be stored in this field. Extended base cost is the cost inclusive of all the taxes that affect the WAC. In case of GTAX , Extended Base Cost = Base Cost + Non-recoverable taxes. In case of VAT, Extended Base Cost = Base Cost.	EXTENDED_BASE_COST	NUMBER(20,4)
INCLUSIVE_COST	Numeric	20,4	N	This will hold the inclusive cost for the primary delivery country of the item. Once a location is associated with the item, the primary locations inclusive cost will be stored in this field. This cost will have both the recoverable and non recoverable taxes included. In case of GTAX , Inclusive Cost = Base Cost + Non-recoverable taxes + Recoverable Taxes. In case of VAT, Inclusive Cost = Base Cost + VAT.	INCLUSIVE_COST	NUMBER(20,4)
BASE_COST	Numeric	20,4	N	This field will hold the tax exclusive cost of the item.	BASE_COST	NUMBER(20,4)

Note: If a record is in the BAD or DISCARD file and the PRIMARY_SUPP_IND is NULL in the file, then the record must be populated with N to be loaded, or the RMS table must be truncated and the entire file must be rerun.

DC_ITEM_SUPP_MANU_COUNTRY.DAT Table

File name: **DC_ITEM_SUPP_MANU_COUNTRY.DAT**

Control file: **DC_ ITEM_ISMC.CTL**

Staging table: **DC_ITEM_SUPP_MANU_COUNTRY**

Suggested post-loading validation:

- Capture counts from ITEM_SUPP_MANU_COUNTRY and compare to flat file DC_ITEM_SUPP_MANU_COUNTRY.DAT to ensure that all rows are loaded.
- Ensure that ITEM_SUPP_MANU_COUNTRY.ITEM is a valid ITEM_MASTER.ITEM.
- Ensure that ITEM_SUPP_MANU_COUNTRY.SUPPLIER is a valid SUPS.SUPPLIER.
- Ensure that ITEM_SUPP_MANU_COUNTRY.MANU_COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	ID of the stock Keeping Unit.	ITEM	VARCHAR2(25)
SUPPLIER	Integer	10	Y	ID of the supplier that supplies the ITEM.	SUPPLIER	NUMBER(10)
MANU_COUNTRY_ID	Alpha-numeric	3	Y	ID of the country where the item is manufactured or originates.	MANU_COUNTRY_ID	VARCHAR2(3)
PRIMARY_MANU_CTRY_IND	Alpha-numeric	1	Y	This field indicates whether this country is the primary country of manufacture for the item/supplier. Each item/supplier combination must have one and only one primary country of manufacture. Valid values are Y or N.	PRIMARY_MANU_CTRY_ID	VARCHAR2(1)

DC_ITEM_SUPP_COUNTRY_DIM Table

File name: **DC_ITEM_SUPP_COUNTRY_DIM.DAT**

Control file: **DC_ISC_DIM.CTL**

Staging table: **DC_ITEM_SUPP_COUNTRY_DIM**

Suggested post-loading validation:

- Capture counts from ITEM_SUPP_COUNTRY_DIM and compare to flat file DC_ITEM_SUPP_COUNTRY_DIM.DAT to ensure that all rows are loaded.
- Ensure that ITEM_SUPP_COUNTRY_DIM.ITEM/SUPPLIER/ ORIGIN_COUNTRY_ID combination exists in ITEM_SUPP_COUNTRY.
- Ensure that ITEM_SUPP_COUNTRY_DIM.DIM_OBJECT is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = DIMO.
- Ensure that ITEM_SUPP_COUNTRY_DIM.PRESENTATION_METHOD is a valid CODE_DETAIL.CODE where CODE_DETAIL.CODE_TYPE = PCKT.
- Ensure that ITEM_SUPP_COUNTRY_DIM.LWH_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS = DIMEN.
- Ensure that ITEM_SUPP_COUNTRY_DIM.WEIGHT_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS = MASS.
- Ensure that ITEM_SUPP_COUNTRY_DIM.LIQUID_VOLUME_UOM is a valid UOM_CLASS.UOM with UOM_CLASS.UOM_CLASS = LVOL.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	ID of the stock keeping unit.	ITEM	VARCHAR2(25)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
SUPPLIER	Integer	10	Y	ID of the supplier that supplies the SKU.	SUPPLIER	NUMBER(10)
ORIGIN_COUNTRY_ID	Alpha-numeric	3	Y	ID of the country in which the item is manufactured or originates.	ORIGIN_COUNTRY_ID	VARCHAR2(3)
DIM_OBJECT	Alpha-numeric	6	Y	Type of dimension object being defined. Valid values exist in the code head/code details tables in RMS in code type DIMO: EA – Each IN – Inner CA – Case PA – Pallet The two-letter code should be included in the file.	DIM_OBJECT	VARCHAR2(6)
PRESENTATION_METHOD	Alpha-numeric	6	N	How the product is presented. Valid values exist in the RMS code head/code details tables with code type PCKT.	PRESENTATION_METHOD	VARCHAR2(6)
LENGTH	Numeric	12,4	N	Length of the packaging used when defining volume. This field is not required but should be populated when the width, height, and LWH_UOM are defined.	LENGTH	NUMBER(12,4)
WIDTH	Numeric	12,4	N	Width of the packaging used when defining volume. This field is not required but should be populated when the length, height, and LWH_UOM are defined.	WIDTH	NUMBER(12,4)
HEIGHT	Numeric	12,4	N	Height of the packaging used when defining volume. This field is not required but should be populated when the length, width, and LWH_UOM are defined.	HEIGHT	NUMBER(12,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
LWH_UOM	Alpha-numeric	4	N	Unit of measure that the length, height, and width values are defined in. This field is not required but should be populated when the length, width, and height are defined. Default from System Options.	LWH_UOM	VARCHAR2(4)
WEIGHT	Numeric	12,4	N	Gross weight of the product. This field is not required but should be populated in conjunction with the following values: NET_WEIGHT WEIGHT_UOM TARE_WEIGHT TARE_TYPE	WEIGHT	NUMBER(12,4)
NET_ WEIGHT	Numeric	12,4	N	Net weight of the product. This field is not required but should be populated in conjunction with the following values: WEIGHT WEIGHT_UOM, TARE_WEIGHT TARE_TYPE	NET_ WEIGHT	NUMBER(12,4)
WEIGHT_UOM	Alpha-numeric	4	N	UOM by which the weight, net weight, and tare weight are defined in. This field is not required but should be populated in conjunction with the following values: WEIGHT NET_WEIGHT TARE_WEIGHT TARE_TYPE Default from System Options.	WEIGHT_UOM	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
LIQUID_VOLUME	Numeric	12,4	N	Liquid volume of the item. This field is not required, but when used, should be used in conjunction with the LIQUID_VOLUME_UOM field.	LIQUID_VOLUME	NUMBER(12,4)
LIQUID_VOLUME_UOM	Alpha-numeric	4	N	Liquid volume unit of measure.	LIQUID_VOLUME_UOM	VARCHAR2(4)
STAT_CUBE	Numeric	12,4	N	Dimensions of the statistical case.	STAT_CUBE	NUMBER(12,4)
TARE_WEIGHT	Numeric	12,4	N	Weight of the tare. This field is not required but should be populated in conjunction with the following values: WEIGHT NET_WEIGHT WEIGHT_UOM TARE_TYPE	TARE_WEIGHT	NUMBER(12,4)
TARE_TYPE	Alpha-numeric	6	N	Indicates whether the tare is considered wet or dry. Valid values are: D - Dry W - Wet This field is not required but should be populated in conjunction with the following values: WEIGHT NET_WEIGHT WEIGHT_UOM TARE_WEIGHT	TARE_TYPE	VARCHAR2(6)

DC_ITEM_COUNTRY Table

File name: **DC_ITEM_COUNTRY.DAT**

Control file: **DC__COUNTRY.CTL**

Staging table: DC_ITEM_COUNTRY

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	ID of the item.	ITEM	VARCHAR2(25)
COUNTRY_ID	Alpha-numeric	3	Y	Contains the unique code that identifies the country.	COUNTRY_ID	VARCHAR2(3)

Required file to load: dc_item_country.dat

DC_ITEM_COST_HEAD Table

File name: **DC_ITEM_COST_HEAD.DAT**

Control file: **DC_ITEM_COST_HEAD.CTL**

Staging table: **DC_ITEM_COST_HEAD**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	ID of the stock keeping unit.	ITEM	VARCHAR2(25)
SUPPLIER	Number	10	Y	ID of the supplier that supplies the SKU.	SUPPLIER	NUMBER(10)
ORIGIN_COUNTRY_ID	Alpha-numeric	3	Y	Country where the item will be sourced from by the supplier.	ORIGIN_COUNTRY_ID	VARCHAR2(3)
DELIVERY_COUNTRY_ID	Alpha-numeric	3	Y	Country to which the item will be delivered to.	DELIVERY_COUNTRY_ID	VARCHAR2(3)
PRIM_DLVY_CTRY_IND	Alpha-numeric	1	Y	Indicates if the country is the primary delivery country of the item.	PRIM_DLVY_CTRY_IND	VARCHAR2(1)
NIC_STATIC_IND	Alpha-numeric	1	Y	Indicates if the Negotiated Item Cost (NIC) is static or not. If NIC is static then the BASE COST of the item will vary based on the location/tax region. If NIC is not static then the NEGOTIATED_ITEM_COST of the item will vary based on the location/tax region.	NIC_STATIC_IND	VARCHAR2(1)
BASE_COST	Number	20,4	Y	This will hold the tax exclusive cost of the item.	BASE_COST	NUMBER(20,4)
NEGOTIATED_ITEM_COST	Number	20,4	Y	This will hold the supplier negotiated item cost.	NEGOTIATED_ITEM_COST	NUMBER(20,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
EXTENDED_BASE_COST	Number	20,4	Y	This will hold the extended base cost of the item. Extended base cost is the cost inclusive of all the taxes that affect the WAC.	EXTENDED_BASE_COST	NUMBER(20,4)
INCLUSIVE_COST	Number	20,4	Y	This will hold the tax inclusive cost of the item. This includes all cost-related taxes - both the recoverable and non-recoverable taxes.	INCLUSIVE_COST	NUMBER(20,4)

Required file to load: dc_item_cost_head.dat

DC_ITEM_COST_DETAIL Table

File name: **DC_ITEM_COST_DETAIL.DAT**

Control file: **DC_ITEM_COST_DETAIL.CTL**

Staging table: **DC_ITEM_COST_DETAIL**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	ID of the stock keeping unit	ITEM	VARCHAR2(25)
SUPPLIER	Number	10	Y	ID of the supplier that supplies the SKU	SUPPLIER	NUMBER(10)
ORIGIN_COUNTRY_ID	Alpha-numeric	3	Y	Country from where the item was sourced.	ORIGIN_COUNTRY_ID	VARCHAR2(3)
DELIVERY_COUNTRY_ID	Alpha-numeric	3	Y	Country to which the item will be delivered to.	DELIVERY_COUNTRY_ID	VARCHAR2(3)
COND_TYPE	Alpha-numeric	10	Y	The condition type applicable on the item's cost. Condition can be a tax code or an expense or a type of a cost of the item.	COND_TYPE	VARCHAR2(10)
COND_VALUE	Number	20,4	N	The condition value or tax amount per of the corresponding condition.	COND_VALUE	NUMBER(20,4)
APPLIED_ON	Number	20,4	N	The value on which given tax is applied.	APPLIED_ON	NUMBER(20,4)
COMP_RATE	Number	20,10	N	The rate of the condition applied.	COMP_RATE	NUMBER(20,10)

Required file to load: dc_item-cost_detail.dat

Load Scripts

DC_ITEM_SUPPLIER.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_SUPPLIER staging table.

LOAD_ITEM_SUPPLIER– This function contains a PL/SQL block that selects from the DC_ITEM_SUPPLIER staging table and inserts the data to the RMS ITEM_SUPPLIER table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ITEM_SUPPLIER to ITEM_SUPPLIER Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	NA
LAST_UPDATE_DATETIME	SYSDATE	NA
PRIMARY_SUPP_IND	N	If NULL Lowest Supplier ID = Y, otherwise default = N Use analytic function. **Note:** The table requires that all records contain PRIMARY_SUPP_IND information, or all records can have this indicator set to NULL.
PALLET_NAME	From SYSTEM_OPTIONS	If NULL
CASE_NAME	From SYSTEM_OPTIONS	If NULL
INNER_NAME	From SYSTEM_OPTIONS	If NULL
CREATE_DATETIME	SYSDATE	NA

Required file to load: dc_item_supplier.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_SUPP_COUNTRY.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_SUPP_COUNTRY staging table.

LOAD_ITEM_SUPP_COUNTRY– This function contains a PL/SQL block that selects from the DC_ITEM_SUPP_COUNTRY staging table and inserts the data to the RMS ITEM_SUPP_COUNTRY table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ITEM_SUPP_COUNTRY to ITEM_SUPP_COUNTRY Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	NA
LAST_UPDATE_DATETIME	sysdate	NA
PRIMARY_SUPP_IND	from item_supplier	NA
PRIMARY_COUNTRY_IND	N	If NULL First Alpha Country ID = Y Otherwise Default = N Use analytic function. The table is required to have all records contain this indicator, or all records can have this indicator set to NULL.
ROUND_LVL	from System Opt.	If NULL.
ROUND_TO_INNER_PCT	from System Opt	If NULL
ROUND_TO_CASE_PCT	from System Opt	If NULL
ROUND_TO_LAYER_PCT	from System Opt	If NULL
ROUND_TO_PALLET_PCT	from System Opt	If NULL
PACKING_METHOD	from System Opt	If NULL

Column Name (RMS Table)	Default Value	Comments
DEFAULT_UOP	Case	If NULL.
LEAD_TIME	from Sups	If NULL
COST_UOM	Standard uom from item_master	If NULL
CREATE_DATETIME	Sysdate	NA
NEGOTIATED_ITEM_COST	The value will be taken from DC_ITEM_COST_HEAD.	NA
EXTENDED_BASE_COST	The value will be taken from DC_ITEM_COST_HEAD.	NA
INCLUSIVE_COST	The value will be taken from DC_ITEM_COST_HEAD.	NA
BASE_COST	The value will be taken from DC_ITEM_COST_HEAD.	NA

Required file to load: dc_item_supp_country.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_SUPP_MANU_COUNTRY.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_SUPP_MANU_COUNTRY staging table.

LOAD_ITEM_SUPP_MANU_COUNTRY– This function should do the following:

Insert the following column values in ITEM_SUPP_MANU_COUNTRY

- item
- supplier
- manu_country_id
- primary_manu_ctry_ind

This function selects from the DC_ ITEM_SUPP_MANU_COUNTRY staging table and inserts the data to the RMS item_supp_manu_country table. All the columns from the staging table defined above will directly map to the RMS table.

Required file to load: dc_item_supp_manu_country.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_COUNTRY.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_COUNTRY staging table.

This function should do the following:

Insert the following columns into the item_country table:

- item
- country_id

This function selects from the dc_item_country staging table and inserts the data to the RMS item_country table. It uses dc_item_country.item = item_master.item and dc_item_country.country_id = country.country_id to join the data to ensure that both the item and the country are valid. All the columns from the staging oracle table defined above will directly map to the RMS table.

Required file to load: dc_item_country.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_COST_HEAD.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_COST_HEAD staging table.

LOAD_ITEM_COST_HEAD– This function should do the following only when default tax type on system options is 'GTAX':

Insert the following columns into the item_cost_head table:

- item
- supplier
- origin_country_id

- delivery_country_id
- prim_dlvy_ctry_ind
- nic_static_ind
- base_cost
- negotiated_item_cost
- extended_base_cost
- inclusive_cost

This function selects from the dc_item_cost_head staging table and inserts the data to the RMS item_cost_head table. It uses dc_item_cost_head.supplier = item_supp_country.supplier and dc_item_cost_head.origin_country_id = item_supp_country.origin_country_id and dc_item_cost_head.item = item_supp_country.item to join the data and to ensure that parent entity ITEM_SUPP_COUNTRY exists. All the columns from the staging table defined above will directly map to the RMS table.

Required file to load: dc_item_cost_head.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_COST_DETAIL.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_COST_DETAIL staging table.

LOAD_ITEM_COST_DETAIL– This function should do the following only when default tax type on system options is 'GTAX':

Insert the following columns into the item_cost_head table:

- item
- supplier
- origin_country_id
- delivery_country_id
- cond_type,
- cond_value,
- applied_on,
- comp_rate

This function selects from the dc_item_cost_head staging table and inserts the data to the RMS item_cost_detail table. It uses dc_item_cost_detail.item = item_cost_head.item and dc_item_cost_detail.supplier = item_cost_head.supplier to join the data and to ensure that the parent entity ITEM_COST_HEAD exists. All the columns from the staging oracle table defined above will directly map to the RMS table.

Required file to load: dc_item_cost_detail.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_PRICE_HIST.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PRICE_HIST staging table.

This function should do the following:

Insert the 0 location record into the RMS price_hist table. Get the unit_cost from the primary supplier and primary country record in the DC_ITEM_SUPP_COUNTRY table for each item.

Get the unit retail, selling uom from DC_PRICE_HIST. You will need to get the primary currency from system options and the supplier's currency from SUPS. If they are different, convert the unit_cost to the primary currency (use one insert/select for records where the supplier currency equals the primary currency (no conversion necessary), use a second for where they are unequal and call CURRENCY_SQL.CONVERT_VALUE).

DC_PRICE_HIST to PRICE_HIST Column Defaults

Column Name (RMS Table)	Default Value	Comments
ACTION_DATE	VDATE	
POST_DATE	VDATE	
SELLING_UNIT_RETAIL	Dc_price_hist.unit_retail	

Required file to load: dc_price_hist.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_SUPP_COUNTRY_DIM.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_SUPP_COUNTRY_DIM staging table.

LOAD_ITEM_SUPP_COUNTRY_DIM– This function contains a PL/SQL block that selects from the DC_ITEM_SUPP_COUNTRY_DIM staging table and inserts the data to the RMS ITEM_SUPP_COUNTRY_DIM table. Most of the columns from the external Oracle table listed above directly map to the RMS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ITEM_SUPP_COUNTRY_DIM to ITEM_SUPP_COUNTRY_DIM Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	N/A
LAST_UPDATE_DATETIME	SYSDATE	N/A
CREATE_DATETIME	SYSDATE	N/A
LWH_UOM	From SYSTEM_OPTIONS	If NULL
WEIGHT_UOM	From SYSTEM_OPTIONS	If NULL

Required file to load: dc_item_supp_country_dim.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_RPM_ITEM_ZONE_PRICE.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_RPM_ITEM_ZONE_PRICE – This function selects from the DC_PRICE_HIST staging table and joins with ITEM_MASTER and RPM_MERCH_RETAIL_DEF to insert data to the RPM RPM_ITEM_ZONE_PRICE table.

The function retrieves the regular zone group ID for the department of the items in the DC_PRICE_HIST table and joins data with the ITEM_MASTER and RPM_MERCH_RETAIL_DEF tables. It performs a bulk collect of this data and loops through the results to insert into the RPM_ITEM_ZONE_PRICE table. For the insert/select, join DC_PRICE_HIST for each item and RPM_ZONE for the department's ZONE_GROUP_ID. The following table indicates the values retrieved for data insert. This function uses the primary currency from the SYSTEM_OPTIONS table. If the zone currency and the primary currency are different, the function converts the UNIT_RETAIL to the zone currency.

DC_PRICE_HIST to RPM_ITEM_ZONE_PRICE Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_ZONE_PRICE_ID	Use sequence	N/A
ITEM	Dc_price_hist.item	N/A
ZONE_ID	Rpm_zone.zone_id	For the department zone_group_id
STANDARD_RETAIL	Dc_price_hist.unit_retail	N/A
STANDARD_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
STANDARD_UOM	Dc_price_hist.uom	N/A
SELLING_RETAIL	Dc_price_hist.unit_retail	N/A
SELLING_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
SELLING_UOM	Dc_price_hist.uom	N/A
MULTI_UNIT_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id

Required file to load: dc_price_hist.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Post-Loading Requirements

After using the data conversion toolset for Item Supplier, you must manually load the ITEM_SUPP_COUNTRY_BRACKET_COST table. This table is required if the supplier has bracket costing.

Manual data loading can be done online through Merchandising applications (RMS or RPM), or you can create scripts. Manual data loading is not included as part of this data conversion toolset. Check with your database administrator to determine the best approach for your data conversion needs.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
```

```
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_item_supplier.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Item Location

This section describes data conversion for the following RMS/RPM tables, listed in the order that they must be loaded:

- ITEM_LOC
- ITEM_LOC_SOH
- RPM_FUTURE_RETAIL
- ITEM_SUPP_COUNTRY_LOC
- FUTURE_COST
- PRICE_HIST

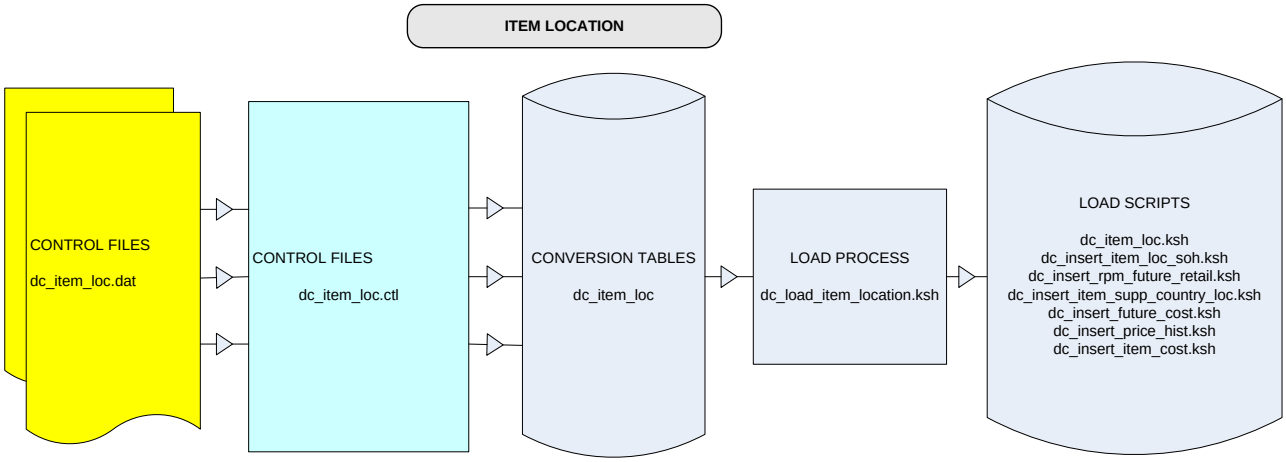
> **Note:** Only data with corresponding RMS ITEM_MASTER records are loaded. Additionally, only items with ITEM_SUPP_COUNTRY data are loaded into the ITEM_SUPP_COUNTY_LOC table.

The following programs are included in this functional area:

- Load Scripts:
  - dc_item_loc.ksh
  - dc_insert_item_loc_soh.ksh
  - dc_insert_rpm_future_retail.ksh
  - dc_insert_item_supp_country_loc.ksh
  - dc_insert_future_cost.ksh
  - dc_insert_price_hist.ksh
  - dc_insert_item_cost.ksh
- Control Files:
  - dc_item_loc.ctl

Data Flow

The following diagram shows the data flow for the Item Location functional area:



Data Flow for the Item Location Functional Area

Prerequisites

Before you begin using the data conversion toolset for Item Location, you must complete data conversion for Items and Item Supplier:

- Fashion Items
- Hardlines
- Grocery Items
- Pack Items
- Item Supplier

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (`^M`) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

STAGING TABLE DEFINITION

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name, Data Type, and length define the physical external table.

DC_ITEM_LOC Table

File name: **DC_ITEM_LOC.DAT**

Control file: **DC_ITEM_LOC.CTL**

Staging table: **DC_ITEM_LOC**

Suggested post-loading validation:

- Ensure that ITEM_SEASONS.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that ITEM_SEASONS.SEASON_ID/PHASE_ID combination exists in PHASES.
- Ensure that ITEM_LOC.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that ITEM_LOC_SOH.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL = ITEM_MASTER.TRAN_LEVEL.
- Ensure that ITEM_LOC.LOC is a valid V_LOCATION.LOCATION_ID with V_LOCATION.STOCKHOLDING_IND = Y.
- Ensure that ITEM_LOC_SOH.ITEM/LOC combination exists on ITEM_LOC.
- Ensure that ITEM_LOC.ITEM_PARENT/ITEM)GRANDPARENT for the item are the same as ITEM_MASTER.ITEM_PARENT, ITEM_GRANDPARENT.
- Ensure that ITEM_LOC.SELLING_UOM is a valid UOM_CLASS.UOM.
- Ensure that ITEM_LOC.PROMO_SELLING_UOM (if not NULL) is a valid UOM_CLASS.UOM.
- Ensure that ITEM_LOC.MULTI_SELLING_UOM (if not NULL) is a valid UOM_CLASS.UOM.
- Ensure that ITEM_LOC.SOURCE_WH is a valid WH.WH where STOCKHOLDING_IND = Y if ITEM_LOC.SOURCE_METHOD = W.
- Ensure that ITEM_LOC.PRIMARY_COST_PACK (if not NULL) is valid ITEM_MASTER.ITEM with ITEM_MASTER.SIMPLE_PACK_IND = Y and that the ITEM_LOC.ITEM = PACKITEM.ITEM when ITEM_LOC.PRIMARY_COST_PACK = PACKITEM.PACK_NO.

> **Note:** If the PRIMARY_LOC_IND field is NULL, any records that are not loaded and are placed in the BAD or DISCARD file must have an N value for this field to rerun. The alternative method is to truncate the RMS table and rerun the entire file.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
SKU	Alpha-numeric	25	Y	Contains the unique identifier for the Stock Keeping Unit (item, product, article)	ITEM	VARCHAR2(25)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
LOCATION	Integer	10	Y	Contains the identifier for the store, warehouse, or external finisher.	LOCATION	NUMBER(10)
LOC_TYPE	Alpha-numeric	1	Y	Defines the type of location. Valid values are: S – store W – warehouse E – external finisher	LOC_TYPE	VARCHAR2(1)
PRIMARY_LOC_IND	Alpha-numeric	1	N	**Note:** Not in the RMS table. This is needed for inserting into item_supp_country_loc. Populate for all item_locs or leave NULL for all item_locs Valid values are Y (to indicate this is the primary location used for inserted into item_supp_country_loc) and N (not the primary location).	PRIMARY_LOC_IND	VARCHAR2(1)
SELLING_UNIT_RETAIL	Numeric	20,4	N	Contains the current selling unit retail for the item/location. This value should contain the current regular unit retail or clearance unit retail but should not reflect any promotional retails. This field is required for sellable items, but not required for non-sellable items. This should be in location currency.	SELLING_UNIT_RETAIL	NUMBER(20,4)
SELLING_UOM	Alpha-numeric	4	N	Contains the unit of measure that the current selling unit retail is defined in. Value values must exist in the RMS UOM_CLASS table and a conversion must exist between the item's standard UOM and the selling UOM. To convert between UOMs in different UOM classes Case type dimensions must be defined at the item/supp/country level for the UOM. This field is required for sellable items, but not required for non-sellable items.	SELLING_UOM	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
TAXABLE_IND	Alpha-numeric	1	N	Indicates if the item is taxable at a store location. Defaults to N when NULL. Any value passed in will be overwritten with an N value for warehouse locations.	TAXABLE_IND	VARCHAR2(1)
LOCAL_SKU_DESC	Alpha-numeric	250	N	May contain a location specific description for the item which differs from the item's primary description.	LOCAL_ITEM_DESC	VARCHAR2(250)
LOCAL_SHORT_DESC	Alpha-numeric	120	N	Contains a shortened location specific description for the item. This field may be used by Point of Sale systems or other systems where display space is limited. This value is defaulted to 120 characters of the local_item_desc when NULL.	LOCAL_SHORT_DESC	VARCHAR2(120)
TI	Numeric	12,4	N	Contains the number of cartons on a layer of a pallet for the item/location (tiers).	TI	NUMBER(12,4)
HI	Numeric	12,4	N	Contains the number of layers on a pallet for the item/location (height).	HI	NUMBER(12,4)
STORE_ORD_MULT	Alpha-numeric	1	Y	This column contains the case pack multiple in which this item needs to be shipped from a warehouse to the location.	STORE_ORD_MULT	VARCHAR2(1)
TICKET_MEAS_OF_EACH	Numeric	12,4	N	Contains the size of an each in terms of the ticketing UOM. For example 12 oz. This value is used in ticketing only.	MEAS_OF_EACH	NUMBER(12,4)
TICKET_MEAS_OF_PRICE	Numeric	12,4	N	Size to be used on the ticket in terms of the ticketing UOM. For example, to have a ticket label print the price per ounce this value would be 1. To show the price per 100 grams this value would be 100. This value is used in ticketing only.	MEAS_OF_PRICE	NUMBER(12,4)
TICKET_UOM	Alpha-numeric	4	N	Unit of measure that will be used on tickets for this item. This value is used in conjunction with the ticket measure of each and ticket measure of price fields.	UOM_OF_PRICE	VARCHAR2(4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
PRIMARY_COST_PACK	Alpha-numeric	25	N	This field contains an item number that is a simple pack containing the item in the item column for this record. If populated, the cost of the future cost table will be driven from the simple pack and the deals and cost changes for the simple pack.	PRIMARY_COST_PACK	VARCHAR2(25)
INBOUND_HANDLING_DAYS	Integer	2	N	Indicates the number of days required to put away or cross dock an item at a warehouse. This value is used for warehouse locations only.	INBOUND_HANDLING_DAYS	NUMBER(2)
SOURCE_WH	Integer	10	N	Required if SOURCE_METHOD = W; null otherwise. This value is used when doing manual store-level replenishment using the inventory request APIs.	SOURCE_WH	NUMBER(10)
SOURCE_METHOD	Alpha-numeric	1	N	Valid values are W (warehouse) or S (supplier). Indicates how inventory for this item is sourced to a store when doing manual store-level replenishment using the inventory request APIs.	SOURCE_METHOD	VARCHAR2(1)
MULT_UNITS	Numeric	12,4	N	If multi-unit pricing is currently being used for this item/location this field will contain the number of qualifying units. For example, if the item is multi-priced as 3 for $25 this field will contain the 3.	MULT_UNITS	NUMBER(12,4)
MULTI_UNIT_RETAIL	Numeric	20,4	N	If multi-unit pricing is currently being used for this item/location this field will contain the multi-retail price. For example, if the item is multi-priced as 3 for $25 this field will contain the $25. This should be in location currency.	MULTI_UNIT_RETAIL	NUMBER(20,4)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
MULTI_SELLING_UOM	Alpha-numeric	4	N	If multi-unit pricing is currently being used for this item/location this field will contain the unit of measure that the multi-unit retail is defined in terms of.	MULTI_SELLING_UOM	VARCHAR2(4)
AVERAGE_WEIGHT	Numeric	12,4	N	This defines the nominal weight for a simple pack catch weight item. Required for a simple pack catch weight item. Null for all others.	AVERAGE_WEIGHT	NUMBER(12,4)
UIN_TYPE	Alpha-numeric	6	N	This column will contain the unique identification number (UIN) used to identify the instances of the item at the location.	UIN_TYPE	VARCHAR2(6)
UIN_LABEL	Alpha-numeric	6	N	This column will contain the label for the UIN when displayed in SIM.	UIN_LABEL	VARCHAR2(6)
CAPTURE_TIME	Alpha-numeric	6	N	This column indicates when the UIN should be captured for an item during transaction processing.	CAPTURE_TIME	VARCHAR2(6)
EXT_UIN_IND	Alpha-numeric	1	Y	Yes/No indicator to indicate whether UIN is being generated in the external system.	EXT_UIN_IND	VARCHAR2(1)

Load Scripts

DC_ITEM_LOC.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_LOC staging table.

LOAD_ITEM_LOC– This function contains a PL/SQL block that selects from the DC_ITEM_LOC staging table and inserts the data to the RMS ITEM_LOC table. It joins the staging table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included. This function performs two inserts, as follows:

- The primary supplier and primary country fields are populated if the item is orderable. First, it populates the RMS ITEM_LOC table with the values from DC_ITEM_LOC joined with a virtual table that selects the primary supplier and the supplier's primary country for the item from THE ITEM_SUPP_COUNTRY table. Also, it joins the table with ITEM_MASTER to get the ORDER_AS_TYPE value for the RECEIVE_AS_TYPE column. This is populated only for buyer packs.
- For the sellable only items, there is no primary supplier or primary country. This is done by limiting the insert to items that do not exist in the RMS ITEM_SUPP_COUNTRY table.

DC_ITEM_LOC to ITEM_LOC Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	N/A
LAST_UPDATE_DATETIME	Sysdate	N/A
TAXABLE_IND	N	If NULL
CLEAR_IND	N	N/A
STORE_PRICE_IND	N	N/A
RPM_IND	N	N/A
LOCAL_SHORT_DESC	rtrim of substrb 120 char of local_item_desc. ITEM_MASTER.SHORT_DESC when local_item_desc is null	If NULL
REGULAR_UNIT_RETAIL	selling_unit_retail	N/A
UNIT_RETAIL	selling_unit_retail	N/A
CREATE_DATETIME	Sysdate	N/A
STATUS_UPDATE_DATE	Sysdate	N/A
STATUS	A	N/A
LOCAL_ITEM_DESC	Default to ITEM_DESC	It will be populated with ITEM_MASTER.ITEM_DESC
RECEIVE_AS_TYPE	ITEM_MASTER.ORDER_AS_TYPE	If item is a buyer pack, pack_type = B and if the location is a warehouse, loc_type = W
ITEM_PARENT	ITEM_MASTER.ITEM_PARENT	N/A
ITEM_GRANDPARENT	ITEM_MASTER.ITEM_GRANDPARENT	N/A

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_ITEM_LOC_SOH.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_LOC_SOH– This function contains a PL/SQL block that selects from the DC_ITEM_LOC staging table and inserts the data to the RMS ITEM_LOC_SOH table. It joins the staging table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included. It joins the staging table with ITEM_MASTER to insert only transactional items (ITEM_LEVEL = TRAN_LEVEL). This function performs two inserts, as follows:

- It joins with RMS ITEM_SUPP_COUNTRY and SUPS tables to get the UNIT_COST and supplier currency, to convert the UNIT_COST into location currency.
- For sellable only items, it does not join with the RMS ITEM_SUPP_COUNTRY and SUPS tables. It creates an insert statement that excludes items that exist in ITEM_SUPP_COUNTRY and sets UNIT_COST to NULL.

The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined):

DC_ITEM_LOC to ITEM_LOC_SOH Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	N/A
LAST_UPDATE_DATETIME	Sysdate	N/A
CREATE_DATETIME	Sysdate	N/A
STOCK_ON_HAND	0	N/A
IN_TRANSIT_QTY	0	N/A
PACK_COMP_INTRAN	0	N/A
PACK_COMP_SOH	0	N/A
TSF_RESERVED_QTY	0	N/A
PACK_COMP_RESV	0	N/A
TSF_EXPECTED_QTY	0	N/A

Column Name (RMS Table)	Default Value	Comments
PACK_COMP_EXP	0	N/A
RTV_QTY	0	N/A
NON_SELLABLE_QTY	0	N/A
CUSTOMER_RESV	0	N/A
CUSTOMER _BACKORDER	0	N/A
PACK_COMP_CUST_RESV	0	N/A
PACK_COMP_CUST_BACK	0	N/A

Column Name (RMS Table)	Default Value	Comments
PACK_COMP_ NON_SELLABLE	0	N/A
ITEM_PARENT	ITEM_MASTER.ITEM_PARENT	N/A
ITEM_GRANDPARENT	ITEM_MASTER.ITEM_GRANDPARENT	N/A
AV_COST	ITEM_SUPP_COUNTRY.unit_cost of the primary supplier/primary country converted to location currency	N/A
PRIMARY_SUPP	ITEM_SUPP_COUNTRY.supplier with primary_supp_ind = Y for item	N/A
PRIMARY_CTRY	ITEM_SUPP_COUNTRY.origin_country_id with primary_supp_ind = Y and primary_country_ind = Y for item	N/A
AVERAGE_WEIGHT	NULL	Only defined if item is a simple pack catch weight item.

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_ITEM_FUTURE_RETAIL.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_FUTURE_RETAIL– This function contains a PL/SQL block that selects from the DC_ITEM_LOC staging table and inserts the data into the RPM RPM_FUTURE_RETAIL table.

> **Note**: Though the INSERT_RPM_FUTURE_RETAIL function can be used to insert data into the RPM_FUTURE_RETAIL table, the suggested approach is to use the seeding logic discussed in the DataConversionSeedFutureRetail Program section.

Many of the columns from the staging table defined above map directly to the RPM table. The exception is to retrieve dept, class, and subclass values for each item from the ITEM_MASTER table. The currency code is retrieved from the STORE or WH table, based on the location and the location type.

The RPM_FUTURE_RETAIL table is loaded for sellable transaction level items only. Even though SELLING_UNIT_RETAIL and SELLING_UOM are not required fields in the DC_ITEM_LOC table, they are required for sellable items. Without the values, inserting into RPM_FUTURE_RETAIL table will fail. Warehouse locations are conditionally inserted into the RPM_FUTURE_RETAIL table, based on the RPM system option RECOGNIZE_WH_AS_LOCATIONS. This uses one insert for stores and checks

this system option before the insert for warehouses. Warehouses must be stockholding locations.

DC_ITEM_LOC to RPM_FUTURE_RETAIL Column Defaults

Column Name (RMS Table)	Default Value	Comments
FUTURE_RETAIL_ID	Sequence	N/A
MULTI_UNIT_RETAIL_ CURRENCY	selling_unit_retail_ currency	Populate if multi_unit_retail is NOT NULL
SELLING_UNIT_RETAIL_ CURRENCY	Lookup store or wh currency	N/A
ACTION_DATE	Vdate	N/A
ZONE_NODE_TYPE	If loc_type = 'S' then 0 If loc_type = 'W' then 2	N/A

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_ITEM_SUPP_COUNTRY_LOC.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_SUPP_COUNTRY_LOC– This function should do the following:

Insert the following column values in RMS ITEM_SUPP_COUNTRY_LOC:

- item
- supplier
- origin_country_id
- loc
- loc_type
- unit_cost
- round_lvl
- round_to_inner_pct
- round_to_case_pct
- round_to_layer_pct
- round_to_pallet_pct
- pickup_lead_time
- create_datetime
- last_update_id
- last_update_datetime
- negotiated_item_cost
- extended_base_cost

- inclusive_cost
- base_cost

The DC_ ITEM_LOC staging Oracle table is joined with the RMS ITEM_SUPP_COUNTRY table and with item_cost_head table to insert data into the RMS ITEM_SUPP_COUNTRY_LOC table for the item's primary supplier/primary country. The function also joins the staging Oracle table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included.

DC_ITEM_LOC to ITEM_SUPP_COUNTRY_LOC Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	N/A
LAST_UPDATE_DATETIME	SYSDATE	N/A
PICKUP_LEAD_TIME	LEAD_TIME	If NULL
CREATE_DATETIME	SYSDATE	N/A
PRIMARY_LOC_IND		If NULL, lowest loc ID = Y, otherwise default = N. Use analytic function. The table requires that all records contain PRIMARY_LOC_IND information, or all records can have this indicator set to NULL.
ROUND_LVL	ITEM_SUPP_COUNTRY. ROUND_LVL	N/A
ROUND_TO_INNER_PCT	ITEM_SUPP_COUNTRY. ROUND_ TO_INNER_PCT	N/A
ROUND_TO_CASE_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_CASE_PCT	N/A
ROUND_TO_LAYER_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_LAYER_PCT	N/A
ROUND_TO_PALLET_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_PALLET_PCT	N/A

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_FUTURE_COST.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_FUTURE_COST– This function selects from the DC_ITEM_LOC staging table, joined with the RMS ITEM_SUPP_COUNTRY_LOC table, and inserts data into the RMS FUTURE_COST table for the item's primary supplier/primary country. Data is inserted into the RMS_FUTURE_COST table for sellable items only.

This function uses the UNIT_COST from the RMS ITEM_SUPP_COUNTRY_LOC table as the value for all the cost columns. It joins the staging table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included.

DC_ITEM_LOC to FUTURE_COST Column Defaults

Column Name (RMS Table)	Default Value	Comments
ACTIVE_DATE	VDATE	N/A
START_IND	Y	N/A
CALC_DATE	VDATE	N/A

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_PRICE_HIST.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_PRICE_HIST– This function should do the following:

Insert the following column values in PRICE_HIST.

- tran_type
- reason
- item
- loc
- loc_type
- unit_cost
- unit_retail
- selling_unit_retail
- selling_uom
- action_date
- multi_units
- multi_unit_retail
- multi_selling_uom

This function selects from the DC_ ITEM_LOC staging table joined with PRICE_HIST for the item's 0 loc record to insert the data to the RMS price_hist table for each item/location combination. The unit_cost is in the primary currency in the 0 PRICE_HIST record so it needs to be converted to local currency. Retrieve the currency_code from the store or wh table based on the location and the loc_type. Retrieve only stockholding warehouses. This function selects from the DC_ITEM_LOC staging table, joined with the RMS PRICE_HIST table for the 0 tran_type, 0 reason, and 0 location record, to insert data into the RMS PRICE_HIST table for each item/location combination.

The UNIT_COST is already in the primary currency for the 0 PRICE_HIST record, so it must be converted to local currency. The function retrieves the CURRENCY_CODE from the RMS STORE or WH table, based on the location and the LOC_TYPE. It retrieves only stockholding warehouses. This function performs the following inserts:

- The location currency (STORE/WH) is equal to the primary currency
- The location currency is different from the primary currency, so it requires the conversion function to convert UNIT_COST.

DC_ITEM_LOC to PRICE_HIST Column Defaults

Column Name (RMS Table)	Default Value	Comments
MULTI_SELLING_UOM	SELLING_UOM	If NULL
SELLING_UNIT_RETAIL	UNIT_RETAIL	If NULL
MULTI_UNIT_RETAIL	UNIT_RETAIL	If NULL
ACTION_DATE	VDATE	N/A
TRAN_TYPE	0	N/A
REASON	0	N/A

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_ITEM_COST.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_COST– This function should do the following when default tax type on system options is 'GTAX':

Insert the following column values in ITEM_COUNTRY

- item
- country_id

This function selects from the DC_ITEM_LOC staging table joined with the ADDR and COUNTRY table and inserts the data to the RMS ITEM_COUNTRY table for the all the locations that the item is ranged to.

Insert the following column values in ITEM_COST_HEAD

- item
- supplier
- origin_country_id
- delivery_country_id
- prim_dlvy_ctry_ind
- nic_static_ind,
- base_cost
- negotiated_item_cost
- extended_base_cost
- inclusive_cost

This function selects from ITEM_SUPP_COUNTRY joined with ITEM_COUNTRY table and inserts the data to the RMS ITEM_COST_HEAD table for the all the locations that the item is ranged to.

Insert the following column values in ITEM_COST_DETAIL

- item
- supplier
- origin_country_id
- delivery_country_id
- cond_type
- cond_value
- applied_on
- comp_rate
- modified_taxable_base

This function selects from the ITEM_COST_HEAD table joined with VAT_ITEM and LOCALIZATION_CONFIG_OPTIONS table and inserts the data to the RMS ITEM_COST_DETAIL table for the all the locations that the item is ranged to.

Required file to load: dc_item_loc.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_item_loc.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

Data Conversion Seed Future Retail Program

The DataConversionSeedFutureRetail program is designed to seed RPM_FUTURE_RETAIL and RPM_ITEM_LOC tables. This will ensure that appropriate data is created for the necessary timelines for RPM to take advantage of rolled up future retail data.

Usage

The following command runs the DataConversionSeedFutureRetail program:

```
dataConversionSeedFutureRetail.sh connect_string load_data slots logpath errpath
```

Arguments

connect_string: The Data Base connection alias

load_data: Load data from data file on to RPM_DC_ITEM_LOC (Y|y|N|n)

slots: The Number of concurrent threads the program should run.

logpath: The directory where the log files would get generated.

errpath: The directory where the error files would get generated.

The OS user that runs the program should have write permissions on the directories specified in the logpath and errpath arguments.

Detail

The DataConversionSeedFutureRetail program seeds RPM_FUTURE_RETAIL and RPM_ITEM_LOC using the data presented in RPM_DC_ITEM_LOC. Users have an option of either populating item-location-selling_retail information on RPM_DC_ITEM_LOC or use this program to load data into RPM_DC_ITEM_LOC from a data file.

In order to load data, the load_data argument should be set to Y or y, the program scans for a file named dc_item_loc.dat in the same directory as the script and loads data from it. The data file should have the following fields delimited by a 1. The number of fields in

the data file is fixed but not all fields need to have values in them. The data loading process would fail even if a single record does not have the required format, the user should correct the record and re run the program. The program deletes all existing records from RPM_DC_ITEM_LOC and loads it with new records from the data file when run with the load_data parameter as Y or y.

> **Note:** This file layout corresponds to the external table DC_ITEM_LOC used by the Merchandising Data Conversion processes.

Data File Layout

Field	Nullable
ITEM	N
LOCATION	N
LOC_TYPE	N
PRIMARY_LOC_IND	Y
SELLING_UNIT_RETAIL	N
SELLING_UOM	N
TAXABLE_IND	Y
LOCAL_ITEM_DESC	Y
LOCAL_SHORT_DESC	Y
TI	Y
HI	Y
STORE_ORD_MULT	Y
MEAS_OF_EACH	Y
MEAS_OF_PRICE	Y
UOM_OF_PRICE	Y
PRIMARY_COST_PACK	Y
INBOUND_HANDLING_DAYS	Y
SOURCE_WH	Y
SOURCE_METHOD	Y
MULTI_UNITS	Y
MULTI_UNIT_RETAIL	Y
MULTI_SELLING_UOM	Y
AVERAGE_WEIGHT	Y

Sample Record:

1234|3|S||23.45|EA|||||||||||||||0|1|EA||

Assumptions

- RPM_FUTURE_RETAIL and RPM_ITEM_LOC are empty before running the program.
- ITEM_MASTER data exists for all items being processed.
- Zone structures have been defined in RPM prior to running this program.
- Merchandize Retail Default Data has been defined in RPM prior to running this program.
- Data will only be created on RPM_FUTURE_RETAIL and RPM_ITEM_LOC by this program.

Primary Tables Involved

- RPM_DC_ITEM_LOC
- RPM_FUTURE_RETAIL
- RPM_ITEM_LOC

Threading

Process Initialization and data loading (if elected) are not threaded. Data processing is threaded at a subclass level. The number of concurrent threads the program executes at a time corresponds to the value entered in the input parameter slots.

> **Note:** If the Unix OS where this batch is executed on is a SunOS, the batch script needs to be manually updated to use the Korn shell interpreter rather than the Bash shell interpreter.

Data Validation

In order to validate data created, using direct counts between RPM_ITEM_LOC and RMS' ITEM_LOC for approved, sellable and transaction level items along with locations recognized by RPM for the dataset staged for this program to process.

Others

This section describes data conversion for the following RMS tables, listed in the order that they must be loaded:

- UDA_ITEM_LOV
- UDA_ITEM_DATE
- UDA_ITEM_FF
- VAT_ITEM (only if the default_tax_type is not GTAX)
- ITEM_SEASONS
- ITEM_TICKET

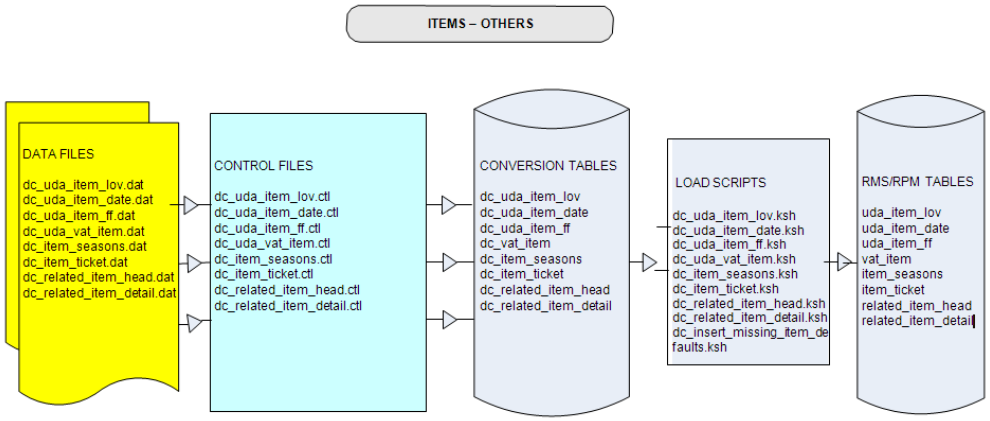
The following programs are included in this functional area:

- Load Scripts:
  - dc_uda_item_lov.ksh
  - dc_uda_item_date.ksh
  - dc_uda_item_ff.ksh
  - dc_vat_item.ksh
  - dc_item_seasons.ksh

  - dc_item_ticket.ksh
  - dc_related_item_head.ksh
  - dc_related_item_detail.ksh
  - dc_insert_missing_item_defaults.ksh
- Control Files:
  - dc_uda_item_lov.ctl
  - dc_uda_item_date.ctl
  - dc_uda_item_ff.ctl
  - dc_vat_item.ctl
  - dc_item_seasons.ctl
  - dc_item_ticket.ctl
  - dc_related_item_head.ctl
  - dc_related_item_detail.ctl

Data Flow

The following diagram shows the data flow for the Items–Others functional area:



Data Flow for the Items – Other Functional Area

Data Flow for the Items – Other Functional Area

Prerequisites

Before you begin using the data conversion toolset for Item Others, you must complete data conversion for Items, Item Supplier, and Item Location:

- Fashion Items
- Hardlines
- Grocery Items
- Pack Items
- Item Supplier
- Item Location

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

STAGING TABLE DEFINITION

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_UDA_ITEM_LOV Table

File name: **DC_UDA_ITEM_LOV.DAT**

Control file: **DC_UDA_ITEM_LOV.CTL**

Staging table: **DC_UDA_ITEM_LOV**

Suggested post-loading validation:

- Ensure that UDA_ITEM_LOV.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that UDA_ITEM_LOV.UDA_ID/UDA_VALUE combination exists in UDA_VALUES.
- Ensure that any UDA_ITEM_LOV.ITEM with a UDA_ITEM_LOV.UDA_ID where UDA.SINGE_VALUE_IND = Y has no other UDA_ITEM_LOV rows.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item associated with the user-defined attribute (UDA). Valid values are any item from the item files: style, SKU, or pack.	ITEM	VARCHAR2(25)
UDA_ID	Integer	5	Y	UDA associated with the item, where the UDA is a list of values (UDA has a DISPLAY_TYPE of LV). Valid values come from the UDA_ID field in the dc_uda.dat file.	UDA_ID	NUMBER(5)

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
UDA_VALUE	Integer	3	Y	List of values value of the UDA. Valid values come from the UDA_VALUE field in the UDA_VALUES table in RMS for the UDA_ID in this file.	UDA_VALUE	NUMBER(3)

DC_UDA_ITEM_DATE Table

File name: **DC_UDA_ITEM_DATE.DAT**

Control file: **DC_UDA_ITEM_DATE.CTL**

Staging table: **DC_UDA_ITEM_DATE**

Suggested post-loading validation:

- Ensure that UDA_ITEM_DATE.ITEM is a valid ITEM_MASTER.ITEM, where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that UDA_ITEM_DATE.UDA_ID is a valid UDA.UDA_ID with UDA.DISPLAY_TYPE of DT.
- Ensure that any UDA_ITEM_DATE.ITEM with a UDA_ITEM_DATE.UDA_ID where UDA.SINGE_VALUE_IND = Y has no other UDA_ITEM_DATE rows.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item associated with UDA. Valid values are any item from the item files: style, SKU, or pack.	ITEM	VARCHAR2(25)
UDA_ID	Integer	5	Y	User-defined attribute associated with the item, where the UDA is a date (UDA has a DISPLAY_TYPE of DT). Valid values come from the UDA_ID field in the dc_uda.dat file.	UDA_ID	NUMBER(5)
UDA_DATE	Date	9	Y	Date value associated with the UDA. Valid values are dates in the date format. Date format is DDMONYYYY (for example, 02JAN2011).	UDA_DATE	DATE

DC_UDA_ITEM_FF Table

File name: **DC_UDA_ITEM_FF.DAT**

Control file: **DC_UDA_ITEM_FF.CTL**

Staging table: **DC_UDA_ITEM_FF**

Suggested post-loading validation:

- Ensure that UDA_ITEM_FF.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that UDA_ITEM_FF.UDA_ID is a valid UDA.UDA_ID with UDA.DISPLAY_TYPE of FF.
- Ensure that any UDA_ITEM_FF.ITEM with UDA_ITEM_FF.UDA_ID, where UDA.SINGE_VALUE_IND = Y, has no other UDA_ITEM_FF rows.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item associated with UDA. Valid values are any item from the item files: style, SKU, or pack.	ITEM	VARCHAR2(25)
UDA_ID	Integer	5	Y	User-defined attribute associated with the item, where the UDA is free-form text (UDA has a DISPLAY_TYPE of FF). Valid values come from the UDA_ID field in the dc_uda.dat file.	UDA_ID	NUMBER(5)
UDA_TEXT	Alpha-numeric	250	Y	Text value associated with the UDA.	UDA_TEXT	VARCHAR2(250)

DC_VAT_ITEM Table

File name: **DC_VAT_ITEM.DAT**

Control file: **DC_VAT_ITEM.CTL**

Staging table: **DC_VAT_ITEM**

Suggested post-loading validation (sequence after dc_load_item_other.ksh) when default tax type is not GTAX (SVAT is used) and will create the DC_VAT_ITEM oracle external table):

- Ensure that VAT_ITEM.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that VAT_ITEM.VAT_REGION is a valid VAT_REGION.VAT_REGION.
- Ensure that VAT_ITEM.VAT_CODE/VAT_RATE is a valid combination in VAT_CODE_RATES, where VAT_ITEM.ACTIVE_DATE >= VAT_CODE_RATES.ACTIVE_DATE, and no other row on VAT_CODE_RATES exists for the combination with a greater ACTIVE_DATE that is still <= VAT_ITEM.ACTIVE_DATE.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item associated with the VAT region. Valid values are any item from the item files: style, SKU, or pack.	ITEM	VARCHAR2(25)
VAT_ REGION	INTEGER	4	Y	Unique identifier of VAT region associated with the item. Valid values come from the VAT_REGION field in the dc_vat_region.dat file.	VAT_REGION	NUMBER(4)
VAT_ TYPE	Alpha-numeric	1	Y	Indicates whether the VAT rate is used for purchasing (cost), selling (retail), or both. Valid values are from the VTTP code type: C, R, or B.	VAT_TYPE	VARCHAR2(1)
VAT_ CODE	Alpha-numeric	6	Y	Unique identifier of value-added tax code, used to determine which items are subject to VAT. Valid values are: S - Standard C - Composite Z - Zero E - Exempt Valid values come from the VAT_CODE column in the dc_vat_codes.dat file.	VAT_CODE	VARCHAR2(6)
VAT_ RATE	Numeric	20,10	Y	Rate of the VAT for the item/ VAT region combination. Valid values come from the VAT_RATE column in the dc_vat_code_rates.dat file. These values exist in the VAT_CODE_RATES table.	VAT_RATE	NUMBER(20,10)
ACTIVE _DATE	Date	9	Y	Date the item/VAT region combination is active. Date format is DDMONYYYY (for example, 02JAN2011).	ACTIVE_DATE	DATE
REVERS E_VAT_I ND	Alpha-numeric	1	Y	Indicates if the Item in the department is subject to reverse charge VAT. Valid values are Y or 'N'.	REVERSE_VAT_IN D	VARCHAR2(1)

DC_ITEM_SEASONS Table

File name: **DC_ITEM_SEASONS.DA**T

Control file: **DC_ITEM_SEASONS.CTL**

Staging table: **DC_ITEM_SEASONS**

Suggested post-loading validation:

- Ensure that ITEM_SEASONS.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that ITEM_SEASONS.SEASON_ID/PHASE_ID combination exists in PHASES.
- Capture count from ITEM_SEASONS and compare to flat file DC_ITEM_SEASONS.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item. Valid values are any item from the item files: style, SKU, or pack.	ITEM	VARCHAR2(25)
SEASON_ID	Integer	3	Y	Identifier of the product season associated to the item. Valid values are from the SEASON_ID field of the SEASONS table in RMS.	SEASON_ID	NUMBER(3)
PHASE_ID	Integer	3	Y	Identifier of the season phase associated with the item. Valid values are from the PHASE_ID field from the PHASES table in RMS, for the given SEASON_ID.	PHASE_ID	NUMBER(3)

Note: If any records are in the BAD or DISCARD file, the RMS table must be truncated and the entire file must be rerun. No new records within a sequence group can be added to the RMS table through the scripts.

DC_ITEM_TICKET Table

File name: **DC_ITEM_TICKET.DAT**

Control file: **DC_ITEM_TICKET.CTL**

Staging table: **DC_ITEM_TICKET**

Suggested post-loading validation:

- Ensure that ITEM_TICKET.ITEM is a valid ITEM_MASTER.ITEM, where ITEM_MASTER.ITEM_LEVEL <=ITEM_MASTER.TRAN_LEVEL.
- Ensure that ITEM_TICKET.TICKET_TYPE_ID is a valid TICKET_TYPE_HEAD.TICKET_TYPE_ID.
- Capture the count from ITEM_TICKET and compare to flat file DC_ITEM_TICKET.DAT to ensure that all rows are loaded.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Item number of item. Valid values are any item from the item files: style, sku, or pack.	ITEM	VARCHAR2(25)
TICKET_TYPE_ID	Alpha-numeric	4	Y	Unique identifier of ticket or label type associated with item. Valid values are from the TICKET_TYPE_ID field in the DC_TICKET_TYPE_HEAD file.	TICKET_TYPE_ID	VARCHAR2(4)
PRINT_ON_PC_IND	Alpha-numeric	1	N	Indicates whether this type of ticket should be printed for this item when a permanent price change goes into effect. Valid values are Y and N. If no value is specified in the file, the value defaults to N.	PRINT_ON_PC_IND	VARCHAR2(1)
PO_PRINT_TYPE	Alpha-numeric	1	N	When the ticket type for the given item should be printed, upon the approval or receipt of the purchase order. Valid values are A and R.	PO_PRINT_TYPE	VARCHAR2(1)
ADDL_OVER_PCT	Numeric	12,4	N	Additional percentage of tickets that should be printed for a given event. For example, if the event is receiving a purchase order, this field holds the percentage of tickets greater than the purchase order quantity that should be printed. If no value is specified in the file, the value defaults to the value from the ticket_over_pct field in the RMS system_options table.	TICKET_OVER_PCT	NUMBER(12,4)

DC_RELATED_ITEM_HEAD Table

File name: **DC_RELATED_ITEM_HEAD.DAT**

Control file: **DC_RELATED_ITEM_HEAD.SQL**

Staging table: **DC_RELATED_ITEM_HEAD**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
RELATIONSHIP_ID	Integer	20	Y	Unique identifier for each relationship header.	RELATIONSHIP_ID	NUMBER(20)
ITEM	Alpha-numeric	25	Y	Item for which the relationships are defined.	ITEM	VARCHAR2 (25)
RELATIONSHIP_NAME	Alpha-numeric	255	Y	Name given to the relationship.	RELATIONSHIP_NAME	VARCHAR2 (255)
RELATIONSHIP_TYPE	Alpha-numeric	6	Y	Describes the type of relationship. Values are configured in code_detail table under code_type IREL.	RELATIONSHIP_TYPE	VARCHAR2(6)
MANDATORY_IND	Alpha-numeric	1	Y	Indicates whether the relationship is mandatory.	MANDATORY_IND	VARCHAR2(1)

DC_RELATED_ITEM_DETAIL Table

File name: **DC_RELATED_ITEM_HEAD.DAT**

Control file: **DC_RELATED_ITEM_DETAIL.SQL**

Staging table: **DC_RELATED_ITEM_DETAIL**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
RELATIONSHIP_ID	Integer	20	Y	Unique identifier for each relationship header.	RELATIONSHIP_ID	NUMBER(20)
RELATED_ITEM	Alpha-numeric	25	Y	Item id of the related item.	RELATED_ITEM	VARCHAR2 (25)
PRIORITY	Integer	4	N	Applicable only in case of relationship type SUBS. In case of multiple related substitute items, this column could be used (optional) to define relative priority.	PRIORITY	NUMBER(4)
EFFECTIVE_DATE	Alpha-numeric	11	N	From this date related item can be used on transactions.	EFFECTIVE_DATE	DATE

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
END_DATE	Alpha-numeric	11	N	Till this date related item can be used on transactions. A value of null means that it is effective forever.	END_DATE	DATE

LOAD SCRIPTS

DC_UDA_ITEM_LOV.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA_ITEM_LOV staging table.

LOAD_UDA_ITEM_LOV– This function contains a PL/SQL block that selects from the DC_UDA_ITEM_LOV staging table and inserts the data to the RMS UDA_ITEM_LOV table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_UDA_ITEM_LOV to UDA_ITEM_LOV Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	sysdate	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	sysdate	Date/time the record was created in RMS. This defaults to the system date.

Required file to load: dc_uda_item_lov.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UDA_ITEM_DATE.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA_ITEM_DATE staging table.

LOAD_UDA_ITEM_DATE– This function contains a PL/SQL block that selects from the DC_UDA_ITEM_DATE staging table and inserts the data to the RMS UDA_ITEM_DATE table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	SYSDATE	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	SYSDATE	Date/time the record was created in RMS. This defaults to the system date.

Required file to load: dc_uda_item_date.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_UDA_ITEM_FF.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_UDA_ITEM_FF staging table.

LOAD_UDA_ITEM_FF– This function contains a PL/SQL block that selects from the DC_UDA_ITEM_FF staging table and inserts the data to the RMS UDA_ITEM_FF table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_UDA_ITEM_FF to UDA_ITEM_FF Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	SYSDATE	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	SYSDATE	Date/time the record was created in RMS. This defaults to the system date.

Required file to load: dc_uda_item_ff.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_VAT_ITEM.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_VAT_ITEM staging table.

LOAD_VAT_ITEM– This function contains a PL/SQL block that selects from the DC_VAT_ITEM staging table and inserts the data to the RMS VAT_ITEM table. If system_options vat_ind is equal to Y and default tax type is NOT 'GTAX' (i.e. 'SVAT' is used), this function selects from the DC_VAT_ITEM and loads directly into the RMS vat_item table. The table below lists columns that do not exist on DC_VAT_ITEM and the defaults to be used for them. If no information is provided in the data file (staging table field values are NULL or not defined).

DC_VAT_ITEM to VAT_ITEM Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	SYSDATE	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	SYSDATE	Date/time the record was created in RMS. This defaults to the system date.
CREATE_ID	Current user id	User who created the record in RMS. This defaults to the Oracle User.
CREATE_DATE	SYSDATE	Date the record was created in RMS. This defaults to the system date.

Required file to load: dc_vat_item.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_SEASONS.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_SEASONS staging table.

LOAD_ITEM_SEASONS– This function contains a PL/SQL block that selects from the DC_ITEM_SEASONS staging table and inserts the data to the RMS ITEM_SEASONS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ITEM_SEASONS to ITEM_SEASONS Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	SYSDATE	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	SYSDATE	Date/time the record was created in RMS. This defaults to the system date.
ITEM_SEASON_SEQ_NO	Sequence generated	Sequence is per item.

Required file to load: dc_item_seasons.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_ITEM_TICKET.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_ITEM_TICKET staging table.

LOAD_ITEM_TICKET– This function contains a PL/SQL block that selects from the DC_ITEM_SEASONS staging table and inserts the data to the RMS ITEM_SEASONS table. The following table defines the default values in the RMS table if no information is provided in the data file (staging table field values are NULL or not defined).

DC_ITEM_TICKET to ITEM_TICKET Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	User who last updated the record in RMS. This defaults to the Oracle User.
LAST_UPDATE_DATETIME	SYSDATE	Date/time the record was last modified in RMS. This defaults to the system date.
CREATE_DATETIME	SYSDATE	Date/time the record was created in RMS. This defaults to the system date.
PRINT_ON_PC_IND	N	If no value is specified in the file, the value defaults to N.
TICKET_OVER_PCT	SYSTEM_OPTIONS. TICKET_OVER_PCT	If no value is specified in the file, the value defaults to the value from the TICKET_OVER_PCT field in SYSTEM_OPTIONS.

Required file to load: dc_item_ticket.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_RELATED_ITEM_HEAD.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_RELATED_ITEM_HEAD staging table.

LOAD_RELATED_ITEM_HEAD– This function selects from the DC_RELATED_ITEM_HEAD and loads directly into the RMS related_item_head table. All the columns are loaded from the staging table itself.

Required file to load: dc_related_item_head.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_RELATED_ITEM_DETAIL.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_RELATED_ITEM_DETAIL staging table.

LOAD_RELATED_ITEM_DETAIL– This function selects from the DC_RELATED_ITEM_DETAIL and loads directly into the RMS related_item_detail table. All the columns are loaded from the staging table itself.

Required file to load: dc_related_item_detail.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

DC_INSERT_ITEM_MISSING_DEFAULTS.KSH

This ksh script will be called to call the load data script to insert data from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

INSERT_ITEM_MISSING_DEFAULTS– This function inserts missing item defaults from the merchandise hierarchy specifications for VAT,UDAs and ITEM Charges. Create 2 cursors to retrieve using bulk collect into a PL/SQL table the ITEM, DEPT, CLASS and SUBCLASS values from ITEM_MASTER.

If vat is turned on in system_options and default tax type is NOT GTAX (i.e. SVAT is used), retrieve sku information and call the VAT_SQL.DEFAULT_VAT_ITEM.

Retrieve style information and call UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL.DC_DEFAULT_CHRGS. Retrieve sku information and call UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL.DEFAULT_CHRGS.

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts is executed. The UNIX administrator can set this by using a script, or the user can

export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_uda_item_lov.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

8

Multiple Sets of Books

This chapter describes the Multiple Sets of Books (MSOB) data conversion. Data must be loaded in this order:

- Partner - Organization Unit
- Transfer Entity – Organization Unit – Set of Books

Prerequisites

Before you begin using the data conversion toolset for Multiple Sets of Books, you must complete data conversion for the Core functional area (dc_load_core.ksh). You also must run the dc_load_partner.ksh script for external finishers for multiple sets of books.

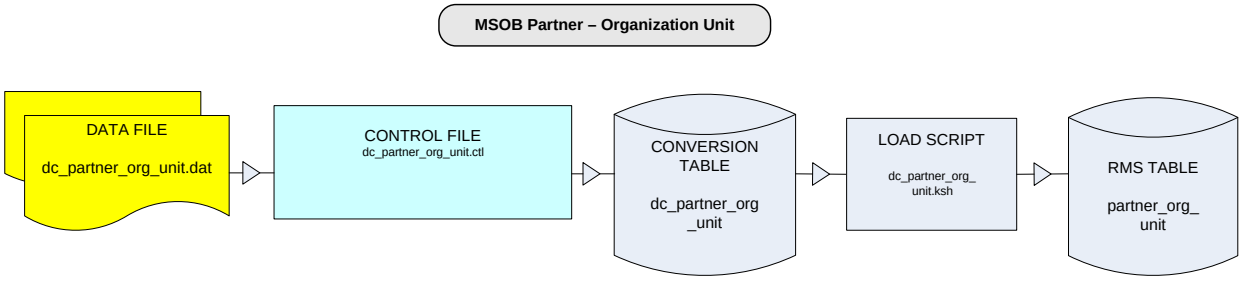
Partner – Organization Unit

This section describes data conversion for the RMS PARTNER_ORG_UNIT table. The following programs are included in this functional area:

- Load script:
  - dc_partner_org_unit.ksh
- Control file:
  - dc_partner_org_unit.ctl

Data Flow

The following diagram shows the data flow for the MSOB Partner – Organization Unit functional area:



Data Flow for MSOB Partner – Organization Unit Functional Area

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

STAGING TABLE DEFINITION

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_PARTNER_ORG_UNIT Table

File name: **DC_PARTNER_ORG_UNIT.DAT**

Control file: **DC_PARTNER_ORG_UNIT.CTL**

Staging table: **DC_PARTNER_ORG_UNIT**

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
PARTNER	Numeric	10	Y	Supplier or Supplier site ID.	PARTNER	VARCHAR2(10)
ORG_UNIT_ID	Numeric	15	Y	Organization Unit ID.	ORG_UNIT_ID	NUMBER(10)
PARTNER_TYPE	Alpha-numeric	1	Y	Type of partner (S for Supplier, U for Supplier Site).	PARTNER_TYPE	VARCHAR2(1)
PRIMARY_PAY_SITE	Alpha-numeric	1	N	Primary payment site indicator.	PRIMARY_PAY_SITE	VARCHAR2(1)

Load Scripts

DC_PARTNER_ORG_UNIT.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_PARTNER_ORG_UNIT staging table.

LOAD_PARTNER_ORG_UNIT– This function contains a PL/SQL block that selects from the DC_PARTNER_ORG_UNIT staging table and inserts the data to the RMS PARTNER_ORG_UNIT table. All the columns from the staging table defined previously map directly to the RMS table.

The following fields are required:

- PARTNER
- ORG_UNIT_ID
- PARTNER_TYPE

Required file to load: dc_partner_org_unit.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts are executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running a Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_partner_org_unit.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

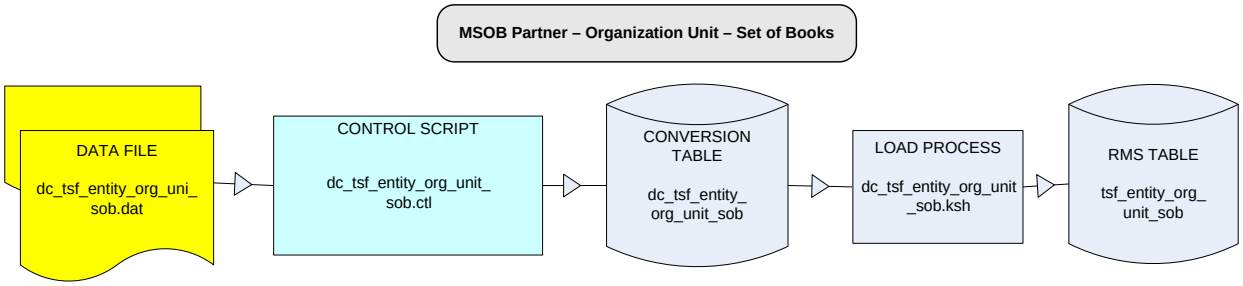
Transfer Entity – Organization Unit – Set of Books

This section describes data conversion for the RMS TFS_ENTITY_ORG_UNIT_SOB table. The following programs are included in this functional area:

- Load script:
  - dc_tsf_entity_org_unit_sob.ksh
- Control file:
  - dc_tsf_entity_org_unit_sob.ctl

Data Flow

The following diagram shows the data flow for the MSOB Transfer Entity – Organization Unit – Set of Books functional area:



Data Flow for the MSOB Transfer Entity – Organization Unit – Set of Books Functional Area

File Format and Staging Tables

The following topics describe the flat file formats that must be created with data from the legacy system. These files must be formatted based on definitions provided before data can be loaded. The data fields for each flat file must be created in the order listed.

File Format

In the table definitions that follow, the File Format columns Field Name, Data Type, and Max Length define the structure of the source file.

> **Note:** Data files must be in UNIX file format and encoded as UTF-8. If a caret-M (^M) can be seen when the file is viewed in a UNIX session, it indicates that the file is in a DOS or Windows format and will cause errors when data is loaded.

Character fields cannot contain carriage returns, because the load process will process a carriage return as an indication of a new record.

STAGING TABLE DEFINITION

In the table definitions that follow, the STAGING TABLE DEFINITION columns Field Name and Data Type (including length) define the physical external table.

DC_TSF_ENTITY_ORG_UNIT_SOB Table

File name: **DC_TSF_ENTITY_ORG_UNIT_SOB.DAT**

Control file: **DC_TSF_ENTITY_ORG_UNIT_SOB.CTL**

Staging table: **DC_TSF_ENTITY_ORG_UNIT_SOB**

Suggested post-loading validation: Ensure that the combination of **TSF_ENTITY_ORG_UNIT_SOB.TSF_ENTITY_ID** and **ORG_UNIT_ID** is unique.

FILE FORMAT					STAGING TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Required**	**Description**	**Field Name**	**Data Type**
TSF_ENTITY_ID	Numeric	10	Y	Transfer Entity ID.	TSF_ENTITY_ID	NUMBER(10)
ORG_UNIT_ID	Numeric	15	Y	Organization Unit ID	ORG_UNIT_ID	NUMBER(15)
SET_OF_BOOKS_ID	Numeric	15	Y	Set of Books ID.	SET_OF_BOOKS_ID	NUMBER(15)

Load Script

DC_TSF_ENTITY_ORG_UNIT_SOB.KSH

This ksh script will be called to serves two purposes:

1. Call SQLLOADER to load flat file data to staging tables and.

2. Call the load data script to insert data from the staging tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the staging tables to the RMS tables.

The following functions should be defined in the declaration of the script:

LOAD_FILE – This function call SQLLOADER to load data from input file to DC_TSF_ENTITY_ORG_UNIT_SOB staging table.

LOAD_TSF_ENTITY_ORG_UNIT_SOB– This function contains a PL/SQL block that selects from the DC_TSF_ENTITY_ORG_UNIT_SOB staging table and inserts the data to the RMS TSF_ENTITY_ORG_UNIT_SOB table. All the columns from the staging table defined previously map directly to the RMS table.

Required file to load: dc_tsf_entity_org_unit_sob.dat

ERROR HANDLING: All functions should include the exception part of the PL/SQL block and handle WHEN OTHERS by assigning the sqlerrm to the KSH variable and return.

COMMIT: Follow each insert statement with a commit command.

Running KSH Scripts

This section describes the preparations for running KSH scripts and the commands to run scripts.

Preparation

Before running a KSH script, ensure that the file has the proper permissions:

```
-rwxrwx-r-x
```

Delete the status (*.status), discard (*.dsc), and bad (*.bad) files.

The environment path variable (PATH) must include the directory where the conversion scripts are executed. The UNIX administrator can set this by using a script, or the user can export the path by doing one of the following (where > represents the UNIX or Linux command line prompt):

Option 1

```
> cd $MMHOME/external/scripts (or the actual script directory)
> export PATH=$PATH:.
```

Option 2

Add the following line to the user .profile file:

```
export PATH=$PATH:$MMHOME/external/scripts (or the actual script directory)
```

Running Script

Run the load script using the following syntax (where > represents the UNIX or Linux command line prompt):

```
> dc_tsf_entity_org_unit_sob.ksh
```

> **Note:** The use of 'ksh' in the command. This prevents the program from exiting the session after it has completed execution.

9

Optional Data

Additional tables can be loaded for each of the functional areas handled by this data conversion toolset. Populating these tables is optional and based on your own business operational needs.

> **Note:** Data conversion for these optional tables must be performed manually. These tables must be loaded after successful conversion of all data as described in the preceding chapters. This is because these optional tables have data referential integrities across functional areas.

The following sections list the optional tables for each of the functional area included in this data conversion toolset. Tables should be loaded in the order that they are listed.

Core Tables

- DIFF_RATIO_HEAD
- DIFF_RATIO_DETAIL
- SOURCE_DLVRY_SCHED
- SOURCE_DLVRY_SCHED_DAYS
- SOURCE_DLVRY_SCHED_EXC
- TRANSIT_TIMES

Merchandise Hierarchy Tables

There is no additional data to be loaded manually.

Organizational Hierarchy Tables

- WH_DEPT
- WH_DEPT_EXPL

Supplier Tables

- SUP_ATTRIBUTES
- SUP_INV_MGMT
- SUP_REPL_DAY
- SUPP_PREISSUE
- SUPS_MIN_FAIL

Items Tables

- PACK_TMPL_HEAD
- PACK_TMPL_DETAIL
- ITEM_SUPP_UOM

- ITEM_LOC_TRAITS
- SUB_ITEMS_HEAD
- SUB_ITEMS_DETAIL
- ITEM_FORECAST
- REPL_ITEM_LOC
- REPL_DAY
- MASTER_REPL_ATTR

A

Appendix: Seed Data Installation

This appendix describes the scripts used to load seed data at the time of installation. The following table outlines data installation scripts supplied by Oracle and the tables populated by these scripts.

> **Note**: Some tables populated by these scripts may be modified for final configuration, or updated with additional values prior to implementation.

Script Name	Scripts/Packages Called	Tables Inserted
RIBDATA.SQL	Calls ALL_RIB_TABLE_VALUES.SQL to insert into the tables.	RIB_ERRORS RIB_LANG RIB_TYPE_SETTINGS RIB_SETTINGS
	Calls RIB_DOCTYPES.SQL, which launches a SQL loader session to insert into the tables.	RIB_DOCTYPES
RMSUOM.SQL	NA	UOM_CLASS
RMSCOUNTRIES.SQL	NA	COUNTRY
RMSSTATES.SQL	NA	STATE
RMSCURRENCIES.SQL	NA	CURRENCIES
STATICIN.SQL	Inserts directly into the tables.	SYSTEM_OPTIONS ADD_TYPE ADD_TYPE_MODULE COST_CHG_REASON DUMMY DEAL_COMP_TYPE DOC_LINK INV_STATUS_TYPES INV_STATUS_CODES LANG MC_REJECTION_REASONS ORDER_TYPES SAFETY_STOCK_LOOKUP TRAN_DATA_CODES TRAN_DATA_CODES_REF TSF_TYPE VEHICLE_ROUND COST_ZONE_GROUP

Script Name	Scripts/Packages Called	Tables Inserted
	Calls ELC_COMP_PRE_HTSUPLD.SQL to insert into the tables.	CVB_HEAD ELC_COMP
	Calls GENERAL_DATA_INSTALL_SQL VAT_CODE_REGION to insert into the tables.	VAT_REGION VAT_CODES VAT_CODE_RATES
	Calls GENERAL_DATA_INSTALL_SQL ADD_TYPE to insert into ADD_TYPE table:	ADD_TYPE
	Calls GENERAL_DATA_INSTALL_SQL ADD_TYPE_MODULE to insert into ADD_TYPE_MODULE table.	ADD_TYPE_MODULE
	Calls GENERAL_DATA_INSTALL. UNIT_OPTIONS to insert into UNIT_OPTIONS table.	UNIT_OPTIONS
	Calls GENERAL_DATA_INSTALL_SQL ELC_COMP_EXPENSES to insert into the tables.	CVB_HEAD CVB_DETAIL
	Calls GENERAL_DATA_INSTALL_SQL ELC_COMP_EXPENSES, GENERAL_DATA_INSTALL_SQL UP_CHARGE and GENERAL_DATA_INSTALL_SQL BACKHAUL_ALLOWANCE to insert into ELC_COMP table.	ELC_COMP
ADD_FILTER_POLICY.SQL	Calls the DBMS_RLS.ADD_POLICY function to implement fine-grained access control.	
CODES.SQL	NA	CODE_HEAD CODE_DETAIL CODE_DETAIL_TL
RESTART.SQL	NA	RESTART_PROGRAM_STATUS RESTART_CONTROL+C11
RTK_ERRORS.SQL	NA	RTK_ERRORS

Script Name	Scripts/Packages Called	Tables Inserted
CONTEXT.SQL	NA	CONTEXT_HELP
UOM_X_ CONVERSION.SQL	NA	UOM_X_CONVERSION
VAR_UPC_EAN_ LOAD.SQL	NA	VAR_UPC_EAN
RMSUOMCONV1.SQL	NA	UOM_CONVERSION
RMSUOMCONV2.SQL	NA	UOM_CONVERSION
CALENDAR.SQL	NA	HALF CALENDAR SYSTEM_VARIABLES PERIOD
SA_SYSTEM_ REQUIRED.SQL	Calls SA_METADATA.SQL to insert into the tables.	POS_TENDER_TYPE_HEAD SA_CC_VAL SA_REFERENCE SA_ERROR_CODES SA_EXPORT_OPTIONS SA_ERROR_IMPACT
	Calls SA_METADATA.SQL, which calls SA_REALM_TYPE.SQL to insert into SA_REALM_TYPE table.	SA_REALM_TYPE
	Calls SA_METADATA.SQL, which calls SA_REALM.SQL to insert into SA_REALM table.	SA_REALM
	Calls SA_METADATA.SQL, which calls SA_PARM_TYPE.SQL to insert into SA_PARM_TYPE table.	SA_PARM_TYPE
	Calls SA_METADATA.SQL, which calls SA_PARM.SQL to insert into SA_PARM table.	SA_PARM
RMS12RTM.SQL	Calls ENTRY_TYPE.SQL to insert into ENTRY_TYPE table.	ENTRY_TYPE
	Calls ENTRY_STATUS.SQL to insert into ENTRY_STATUS table.	ENTRY_STATUS
	Calls OGA.SQL to insert into OGA table.	OGA
	Calls TARIFF_TREATMENT.SQL to insert into TARIFF_TREATMENT table.	TARIFF_TREATMENT
	QUOTA_CATEGORY.SQL	QUOTA_CATEGORY

Script Name	Scripts/Packages Called	Tables Inserted
	Calls COUNTRY_TARIFF_ TREATMENT.SQL to insert into COUNTRY_TARIFF_ TREATMENT table.	COUNTRY_TARIFF_ TREATMENT
	Calls HTS_HEADINGS.SQL to insert into HTS_CHAPTER table.	HTS_CHAPTER