Oracle® Retail Merchandising

Data Conversion Operations Guide
Release 13.2.1

April 2011

ORACLE

Oracle® Retail Merchandising Data Conversion Operations Guide, Release 13.2.1

Primary Author: Susan McKibbon

Contributors: Zhenzhen Stein

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

Contents

Send Us Your Comments

Oracle Retail Merchandising, Data Conversion Operations Guide, Release 13.2.1

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

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

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

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

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

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

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

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

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

Preface

This 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.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at http://www.oracle.com/accessibility/.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Audience

This guide is for:

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

Related Documents

For more information, see the following documents in the Oracle Retail Merchandising Release 13.2.1 documentation set:

- *Oracle Retail Merchandising Batch Sch**edule***
- Oracle Retail Fiscal Management documentation set
- 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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 13.2) or a later patch release (for example, 13.2.1). If you are installing the base release and additional patch and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

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/technology/documentation/oracle_retail.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

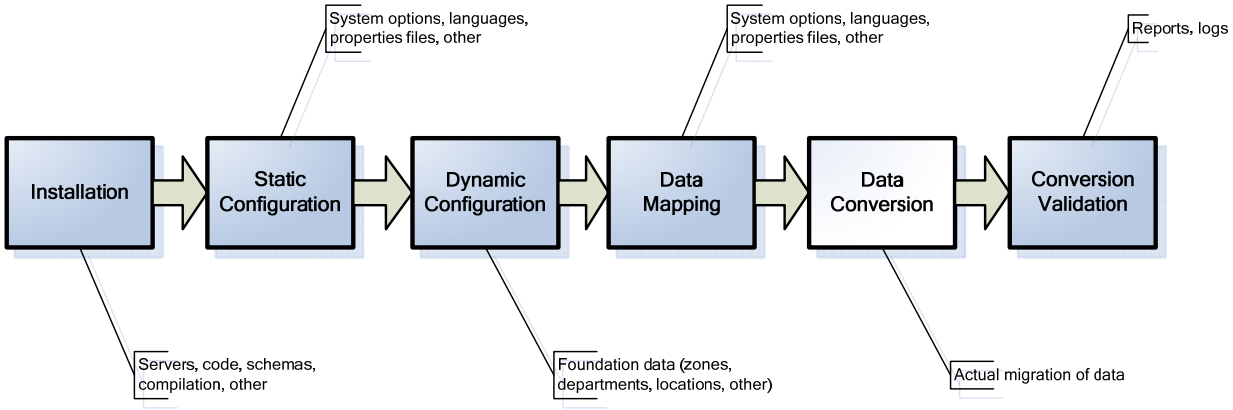
Data Conversion Overview

This chapter is a brief introduction to the overall process to convert legacy data to the tables required by the Oracle Retail Merchandising applications. You 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.



Before actual data conversion can begin, 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 members perform 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 input files provided by the legacy system. These scripts are executed in sequence through a master wrapper script (UNIX shell script). 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:

- Master script DC_LOAD_MAIN.KSH

  This is the master script that starts the segment load scripts for each functional area. This KSH script can run other KSH or SQL scripts that are not in covered in the Oracle Retail Merchandising conversion toolset, such as future/custom validation scripts. You can customize this script, and you can configure it to load any specified number of tables at one time.

- Segment load scripts

  For each functional area, the segment load scripts build the external tables by calling the database conversion table create scripts (SQL) and load data from the flat files by calling the SQL load script. The segment load script can also be configured to load only the data, skipping the table creation.

- External Oracle table create scripts

  For each functional area, the database conversion table create scripts contain the table column definitions, the target data file (*.dat file), the Oracle directory where the data files are located, and the location where the bad, discard, and log files are created. The load script has internal functions that select from the external tables and insert into Oracle Merchandising tables.

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

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 is assumed to be clean data. There is no data validation included in this toolset.
- Database constraints should 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:

- The master script used to run the auto-loading process
- The 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
- Brazil Localization

> **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	Master Script (DC_LOAD_MAIN.KSH)	This chapter describes the master script, the main tool used to run the auto-loading process. It 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 (MSOB)	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.
10	Brazil Localization	This chapter describes in detail the programs and files required to load data for each of the Brazil localization areas.

2

Master Script (DC_LOAD_MAIN.KSH)

The configurable master script executes the data load for each functional area in the data conversion toolset. It feeds from a sequence file (*.seq) that lists other scripts or executable programs used to load data to the RMS tables. The script can execute the following:

- Other Korn shell (ksh) scripts (segment wrapper scripts to load data to RMS tables)
- SQL scripts (to provide custom validation or table counts after loads)
- Single- or multi-threaded RMS batch programs (to support conversion of functional areas such as organization; for example, running storeadd)

The sequence file allows you to customize the execution of programs. You can customize the master script, and you can configure it to load any specified number of tables at one time. The script can be configured to run each segment wrapper script independently, allowing you to load smaller sets of data.

> **Note:** Although the master script allows you to run all functional areas in succession, it is not intended for this purpose. There are prerequisites and post-loading requirements for each functional area, as well as other dependencies across functional areas. Refer to Chapters 3 through 7 for additional details about loading requirements.

Common functions such as error handling and messaging (writing information to log files) are handled in a separate library file called in the dc_load_main.ksh script. Global variables, script, data, and other directories are defined in a separate configuration file.

Because most clients will be loading data into the Oracle tables with constraints disabled, it is advisable to add custom SQL scripts for validation. Calls to these SQL scripts can be added to the sequence file after the load script. Suggested validation is noted in each section, to assist in developing and adding validation scripts to ensure a successful conversion. These validations will be done after the individual load completes and prior to enabling the constraints. The nature and quantity of load issues indicated by the load validations will help determine, along with analysis of the log files, whether a reload should be considered.

Configuration File Definition (DC_LOAD.CFG)

The configuration file contains all directory paths for the executable KSH scripts, SQL scripts, and so on. It also contains log and temp directories needed by the dc_load_main.ksh script and the segment 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. See Appendix: Seed Data Installation for more information.

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
Oracle data directory	orclDataDir	This is the directory name defined within the database that points to a system directory that contains the data files used by the external tables (dataDir).
Oracle log directory	orclLogDir	This is the directory name defined within the database that points to a system directory that contains the log files generated by the external tables. **Note:** In some instances, an entry in the external table log may be repeated several times, because the external table may be used in several inserts.
Data directory	dataDir	This directory contains the data files (*.dat) used to load information to the external Oracle 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 used in the conversion process.
Log directory	logDir	This directory contains the conversion script execution logs. **Note:** This directory is different from the orclLogDir. The logDir contains the daily logs generated by the conversion scripts. The orclLogDir contains logs generated by the external tables.
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.

Sequence File Definition

The sequence file (*.seq) is a file that lists executables to be run in sequence. This file can specify KSH scripts, SQL scripts, RMS batch programs, or other executables. The dc_load_main.ksh script reads from this file, so this file must be configured prior to running the master script.

The lines of the file have the following format:

```
<tag> <script-name> <script-parameters>
```

Tags are as follows:

Tag	Description
PGM	Use this tag to run KSH scripts or other executable scripts or applications. If no path is defined with the script name, it is assumed that the script resides in the script directory defined in the configuration file (*.cfg).
SQL	Use this tag to run SQL scripts. If no path is included with the script name, it is assumed that the script resides in the script directory defined in the configuration file.
BRPGM	Used to run Brazil specific ksh scripts or other executable file. If no path is included with the script name, it assumes the script resides in the script directory defined in the configuration file.
RMS	Use this tag to run RMS batch programs. Although these are executable, RMS tables must be referenced to automatically thread the execution. It is assumed that the batch executable is located in the bin directory unless specified otherwise (when using customized programs) and has an entry in the restart control tables (except for prepost).
>	Use this tag before text lines to display custom messages to both the log file and screen.
#	Use this tag before text lines to include remarks in the sequence file that are ignored during execution.

Example:

```
# This section will load the supplier information to the RMS tables.
> Running LOAD_SUPPLIER.KSH…
PGM load_supplier.ksh
> Loading supplier information completed.
# Now validate if the data is loaded successfully.
SQL dc_validate_supplier.sql
```

> **Note:** The sequence file should terminate with a new line. The tag, program name, and parameters are all separated by spaces.

Library File Description (DC_LOAD.LIB)

The dc_load.lib file is a collection of common functions used by the dc_load_main.ksh and segment 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) and sequence (*.seq) 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, as read from the *.seq file. 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 subprocess (& appended to the batch execution line).
- **execBRPgm**—This function is called from the main script to execute KSH or other executable scripts, as read from the *.seq file. 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.

Master Script Technical Flow

The dc_load_main.ksh script first calls the dc_load.cfg configuration file and the dc_load.lib library file to set up the environment variables, such as directories, and to define common functions. Most of the actual logic resides in the library.

The main section first validates whether the sequence file passed in is valid (if it exists, is readable, and has data).

If the file is valid, the script proceeds to read the information from the file line by line:

- The script assumes that the first value it reads is the tag value and stores this value in the tag variable.
- The second value contains the program name (program name only, or with specific path names for files not in the script directory defined in the configuration file). This information is stored in the seqData1 variable.
- The third value contains the rest of the program parameters and is stored in the seqData2 variable.

Using a case statement, the script checks each tag and executes the appropriate command or function:

- Empty lines, and those with the # tag, are ignored.
- Text lines after the > tag are displayed to the screen and written to the log file. This tag is used to indicate the current processing point.
- For lines with the PGM tag, the script calls the execPgm library function.
- Similarly for lines with BRPGM, SQL and RMS, the script calls the execBRPgm, execSql and execRms library functions, respectively.

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 will be 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 master script using the following syntax (where **>** represents the UNIX or Linux command line prompt):

```
> dc_load_main.ksh –q <sequence-file-name>
```

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

To run individual segment wrapper scripts, the '–q <*sequence-file-name*>' portion of the command line is not required. For example:

```
> dc_load_core.ksh
```

Note: When the KSH script calls SQL scripts to load external tables, it is common to encounter the following error. This is because there may not be an external Oracle table to DROP if the table does not exist in the database. No additional action is required.

```
ERROR at line 1:
ORA-00942: table or view does not exist
```

The data loading process does not truncate RMS tables; it only DROPs external Oracle tables. The KSH script can be used to load data to the same table in multiple phases.

3

Core

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

- CUSTOMER
- 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

The following programs are included in this functional area:

- Main wrapper script dc_load_main.ksh

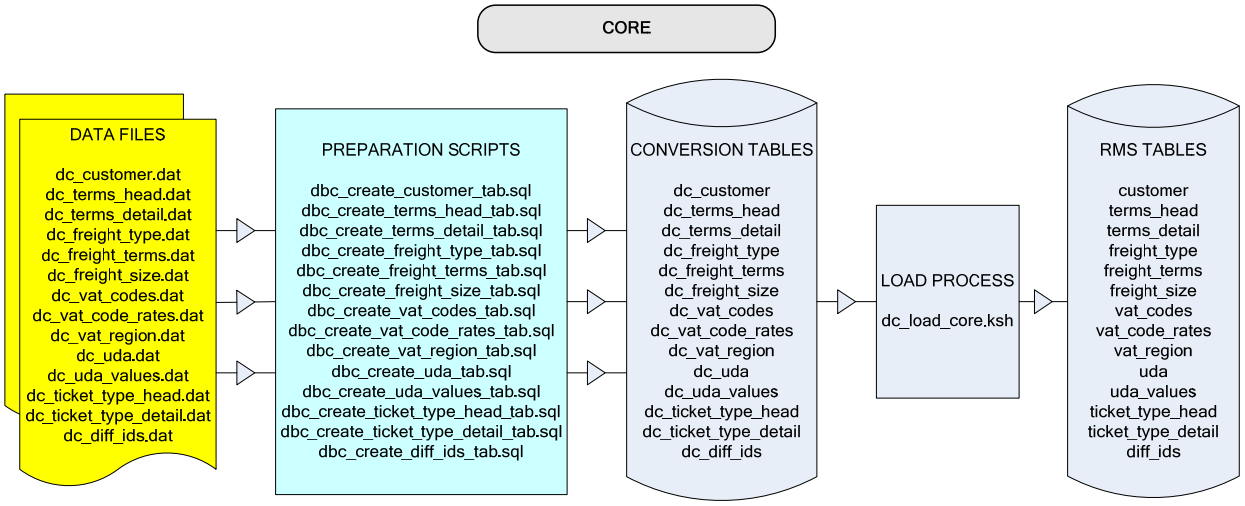
  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_core.ksh

  This wrapper calls the external Oracle table create and load scripts.
- External Oracle table create scripts:
  - dbc_create_customer_tab.sql
  - dbc_terms_head_tab.sql
  - dbc_terms_detail_tab.sql
  - dbc_create_freight_type_tab.sql
  - dbc_create_freight_terms_tab.sql
  - dbc_create_freight_size_tab.sql
  - dbc_create_vat_codes_tab.sql
  - dbc_create_vat_code_rates_tab.sql
  - dbc_create_vat_region_tab.sql
  - dbc_create_uda_tab.sql
  - dbc_create_uda_values_tab.sql
  - dbc_create_ticket_type_head_tab.sql

- dbc_create_ticket_type_detail_tab.sql
- dbc_create_diffids_tab.sql
- dbc_create_tsf_entity_tab.sql
- dbc_create_fif_gl_setup_tab.sql
- dbc_create_org_unit_tab.sql

Data Flow

The following diagram shows the data flow for the 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. See Appendix: Seed Data Installation for more information.

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

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

Table	Required / Optional / Comments
TSF_ENTITY	Required when MLE is turned on
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 upcharges will be loaded
STATE	Required only if using addresses in U.S. locations

File Format and External Oracle 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_core.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.

External Oracle Table Definition

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

Customer—DC_CUSTOMER Table

File name: **DC_CUSTOMER.DAT**

Table create SQL script: **DBC_CREATE_CUSTOMER_TAB.SQL**

External Oracle table created: **DC_CUSTOMER**

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

- Ensure that CUSTOMER.CUST_ID is unique.
- Ensure that CUSTOMER.CUST_STATE is a valid STATE.STATE.
- Ensure that CUSTOMER.CUST_COUNTRY_ID is a valid COUNTRY.COUNTRY_ID.
- Capture the count from CUSTOMER and compare to flat file DC_CUSTOMER.DAT to ensure that all rows are loaded.

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req Ind**	**Description**	**Field Name**	**Data Type**
CUST_ID	Alpha-numeric	10	Y	Code that uniquely identifies the customer.	CUST_ID	VARCHAR2(10)
CUST_NAME	Alpha-numeric	120	Y	Customer name.	CUST_NAME	VARCHAR2(120)
CONTACT_ NAME	Alpha-numeric	120	N	Contact name, if different from the customer name.	CONTACT_ NAME	VARCHAR2(120)
CUST_TITLE	Alpha-numeric	250	N	Title associated with customer (for example, Mr., Mrs., Dr.).	CUST_TITLE	VARCHAR2(250)
CUST_ADD1	Alpha-numeric	240	Y	Customer street et address line 1.	CUST_ADD1	VARCHAR2(240)
CUST_ADD2	Alpha-numeric	240	N	Customer street address line 2. Optional.	CUST_ADD2	VARCHAR2(240)
CUST_CITY	Alpha-numeric	120	Y	Customer address city.	CUST_CITY	VARCHAR2(120)
CUST_STATE	Alpha-numeric	3	N	Customer address state. This must be a valid state in the STATE table.	CUST_STATE	VARCHAR2(3)
CUST_ COUNTRY_ ID	Alpha-numeric	3	Y	Customer address country ID.	CUST_ COUNTRY_ ID	VARCHAR2(3)
CUST_POST	Alpha-numeric	30	N	Customer address postal code.	CUST_POST	VARCHAR2(30)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
DAY_PHONE	Alpha-numeric	20	Y	Customer daytime telephone number.	DAY_PHONE	VARCHAR2(20)
EVE_PHONE	Alpha-numeric	20	N	Customer evening telephone number.	EVE_PHONE	VARCHAR2(20)

Terms

DC_TERMS_HEAD Table

File name: **DC_TERMS_HEAD.DAT**

Table create SQL script: **DBC_CREATE_TERMS_HEAD_TAB.SQL**

External Oracle table created: **DC_TERMS_HEAD**

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

- 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					EXTERNAL ORACLE 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					EXTERNAL ORACLE TABLE DEFINITION	
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**

Table create SQL script: **DBC_CREATE_TERMS_DETAIL_TAB.SQL**

External Oracle table created: **DC_TERMS_DETAIL**

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

- 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					EXTERNAL ORACLE 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_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,)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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 first 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 the month used to calculate discount date of invoice payment line. For example, 1 represents the first 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	9	N	Fixed due date. Date format is DDMONYYYY (for example, 02JAN2011).	FIXED_DATE	DATE

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
START_ DATE_ ACTIVE	Alpha-numeric	9	N	Date in which the payment terms become active Date format is DDMONYYYY.	START_DATE_ ACTIVE	DATE
END_DATE_AC TIVE	Alpha-numeric	9	N	Date in which the payment terms become inactive Date format is DDMONYYYY.	END_DATE_ ACTIVE	DATE

Freight

DC_FREIGHT_TYPE Table

File name: **DC_FREIGHT_TYPE.DAT**

Table create SQL script: **DBC_CREATE_FREIGHT_TYPE_TAB.SQL**

External Oracle table created: **DC_FREIGHT_TYPE**

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

- 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					EXTERNAL ORACLE 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_T YPE	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**

Table create SQL script: **DBC_CREATE_FREIGHT_TERMS_TAB.SQL**

External Oracle table created: **DC_FREIGHT_TERMS**

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

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_FREIGHT_SIZE_TAB.SQL**

External Oracle table created: **DC_FREIGHT_SIZE**

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

- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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**

Table create SQL script: **DBC_CREATE_VAT_CODES_TAB.SQL**

External Oracle table created: **DC_VAT_CODES**

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

- 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					EXTERNAL ORACLE 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. Valid values are: S - Standard C - Composite Z - Zero E- Exempt	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**

Table create SQL script: **DBC_CREATE_VAT_CODE_RATES_TAB.SQL**

External Oracle table created: **DC_VAT_CODE_RATES**

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

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_VAT_REGION_TAB.SQL**

External Oracle table created: **DC_VAT_REGION**

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

- 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					EXTERNAL ORACLE 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)
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)

UDA

DC_UDA Table

File name: **DC_UDA.DAT**

Table create SQL script: **DBC_CREATE_UDA_TAB.SQL**

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

- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
DISP LAY_ TYPE	Alpha-numeric	2	Y	How the UDA will be displayed 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_DESCRIPT ION. For FF, this constrains what is stored in UDA_ITEM_FF. UDA_TEXT.	DATA_ LENGTH	VARCHAR2(3)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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**

Table create SQL script: **DBC_CREATE_UDA_VALUES_TAB.SQL**

External Oracle table created: **DC_UDA_VALUES**

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

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_TICKET_TYPE_HEAD_TAB.SQL**

External Oracle table created: **DC_TICKET_TYPE_HEAD**

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

- 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					EXTERNAL ORACLE 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)
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**

Table create SQL script: **DBC_CREATE_TICKET_TYPE_DETAIL_TAB.SQL**

External Oracle table created: **DC_TICKET_TYPE_DETAIL**

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

- 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					EXTERNAL ORACLE 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)
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**

Table create SQL script: **DBC_CREATE_DIFF_IDS_TAB.SQL**

External Oracle table created: **DC_DIFF_IDS**

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

- 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					**EXTERNAL ORACLE 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)
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 _SUB GROUP	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**

Table create SQL script: **DBC_CREATE_TSF_ENTITY_TAB.SQL**

External Oracle table created: **DC_TSF_ENTITY**

Suggested post-loading validation (sequence after dc_load_core.ksh): Ensure that **TSF_ENTITY.TSF_ENTITY_ID** is unique.

FILE FORMAT					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_FIF_GL_SETUP_TAB.SQL**

External Oracle table created: **DC_FIF_GL_SETUP**

Suggested post-loading validation (sequence after dc_load_core.ksh): Ensure that **FIF_GL_SETUP.SET_OF_BOOKS_ID** is unique.

FILE FORMAT					EXTERNAL ORACLE 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	Alpha-Numeric	15	Y	Last updated ID for transactions.	TSF_ENTITY_ID	VARCHAR2(15)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
SEQUENCE 10_DESC	Alpha-Numeric	20	N	Contains description for sequence columns on the interface cross-reference form.	SEQUENCE10_DESC	VARCHAR2(20)
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.	CURRENCY_CODE	VARCHAR2(3)

Organization Unit—DC_ORG_UNIT Table

File name: **DC_ORG_UNIT.DAT**

Table create SQL script: **DBC_CREATE_ORG_UNIT_TAB.SQL**

External Oracle table created: **DC_FIF_GL_SETUP**

ORG_UNIT

Suggested post-loading validation (sequence after dc_load_core.ksh): Ensure that **ORG.UNIT.ORG_UNIT_ID** is unique.

FILE FORMAT					EXTERNAL ORACLE 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)

DC_LOAD_CORE.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create the external Oracle tables.
- It calls the load data script to insert data from the external Oracle tables to the RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_core.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_CUSTOMER

This function contains a PL/SQL block that selects from the DC_ CUSTOMER external table and inserts the data to the RMS CUSTOMER table. All the columns from the external Oracle table defined previously map directly to the RMS table. The fields that are not required are NULL.

The function returns a Boolean value.

Required file to load: dc_customer.dat

LOAD_TERMS_HEAD

This function contains a PL/SQL block that selects from the DC_ TERMS_HEAD external table and inserts the data to the RMS TERMS_HEAD table. All the columns from the external Oracle table defined previously map directly to the RMS table. The fields that are not required are NULL.

The function returns a Boolean value.

Required file to load: dc_terms_head.dat

LOAD_TERMS_DETAIL

This function contains a PL/SQL block that selects from the DC_ TERMS_DETAIL external table and inserts the data to the RMS TERMS_DETAIL table. All the columns from the external Oracle table defined previously map directly to the RMS table. The fields that are not required are NULL.

The function returns a Boolean value.

Required files to load: dc_terms_head.dat, dc_terms_detail.dat

LOAD_FREIGHT_TYPE

This function contains a PL/SQL block that selects from the DC_FREIGHT_TYPE external table and inserts the data to the RMS FREIGHT_TYPE table. All the columns from the external Oracle table defined previously map directly to the RMS table. All are required.

The function returns a Boolean value.

Required file to load: dc_freight_type.dat

LOAD_FREIGHT_TERMS

This function contains a PL/SQL block that selects from the DC_FREIGHT_TERMS external table and inserts the data to the RMS FREIGHT_TERMS table. All the columns from the external Oracle table defined previously map directly to the RMS table.

The function returns a Boolean value.

Required file to load: dc_freight_terms.dat

LOAD_FREIGHT_SIZE

This function contains a PL/SQL block that selects from the DC_FREIGHT_SIZE external table and inserts the data to the RMS FREIGHT_SIZE table. All the columns from the external Oracle table defined previously map directly to the RMS table. All are required.

The function returns a Boolean value.

Required file to load: dc_freight_size.dat

LOAD_VAT_CODES

This function contains a PL/SQL block that selects from the DC_VAT_CODES external 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 (SVAT is used). All the columns from the external oracle table defined above will directly map to the RMS table and are required.

The function returns a Boolean value.

Required file to load: dc_vat_codes.dat

LOAD_VAT_CODE_RATES

This function contains a PL/SQL block that selects from the DC_VAT_CODE_RATES external 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 (SVAT is used). All the columns from the external oracle 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 (external table field value is NULL or not defined).

The function returns a Boolean value.

DC_VAT_CODE_RATES to VAT_CODE_RATES Column Defaults

Field Name (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 files to load: dc_vat_codes.dat (required to verify if VAT_CODES was loaded previously), **dc_vat_code_rates.dat**

LOAD_VAT_REGION

This function contains a PL/SQL block that selects from the DC_VAT_REGION external 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 (SVAT is used.). All the columns from the external oracle table defined above will directly map to the RMS table and all are required.

The function returns a Boolean value.

Required file to load: dc_vat_region.dat

LOAD_UDA

This function contains a PL/SQL block that selects from the DC_UDA external table and inserts the data into the RMS UDA table. All the columns from the external Oracle table defined previously 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 (external table field values are NULL or not defined).

The function returns a Boolean value

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.

Field Name (RMS Table)	Default Value	Comments
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

LOAD_UDA_VALUES

This function contains a PL/SQL block that selects from the DC_UDA_VALUES external 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 external Oracle table defined previously 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 (external table field values are NULL or not defined).

The function returns a Boolean value

DC_UDA_VALUES to UDA_VALUES Column Defaults

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

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

LOAD_TICKET_TYPE_HEAD

This function contains a PL/SQL block that selects from the DC_TICKET_TYPE_HEAD external table and inserts the data into the RMS TICKET_TYPE_HEAD table. All the columns from the external Oracle table defined previously map directly to the RMS table.

The function returns a Boolean value.

Required files to load: dc_ticket_type_head.dat

LOAD_TICKET_TYPE_DETAIL

This function contains a PL/SQL block that selects from the DC_TICKET_TYPE_DETAIL external table and inserts the data into the RMS TICKET_TYPE_DETAIL table. All the columns from the external Oracle table defined previously 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 (external table field values are NULL or not defined). There should be a value in TICKET_ITEM_ID or UDA_ID, but not both.

The function returns a Boolean value.

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

LOAD_DIFF_IDS

This function contains a PL/SQL block that selects from the DC_DIFF_IDS external table and inserts the data to the RMS DIFF_IDS table. All the columns from the external Oracle table defined previously map directly to the RMS table. All are required.

The following table defines the default values in the RMS table if no information is provided in the data file (external table field values are NULL or not defined). There should be a value in TICKET_ITEM_ID or UDA_ID, but not both.

The function returns a Boolean value.

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

LOAD_TSF_ENTITY

This function contains a PL/SQL block that selects from the DC_TSF_ENTITY external table and inserts the data to the RMS TSF_ENTITY table. All the columns from the external Oracle table defined previously map directly to the RMS table. All fields are required.

The function returns a Boolean value.

Required file to load: dc_tsf_entity.dat

LOAD_FIF_GL_SETUP

This function contains a PL/SQL block that selects from the DC_FIF_GL_SETUP external table and inserts the data to the RMS FIF_GL_SETUP table. All the columns from the external Oracle table defined previously map directly to the RMS table. SET_OF_BOOKS_ID.LAST_UPDATE_ID, CATEGORY_ID, DELIVER_TO_LOCATION_ID, DESTINATION_ORGANIZATION_ID, SET_OF_BOOKS_ID, and CURRENCY_CODE are required fields.

The function returns a Boolean value.

Required file to load: dc_fif_gl_setup.dat

LOAD_ORG_UNIT

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

The function returns a Boolean value.

Required file to load: dc_org_unit.dat

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

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 system_options.vat_ind is Y and default_tax_type is not GTAX)
- UDA_ITEM_DEFAULTS
- RPM_DEPT_AGGREGATION

The following programs are included in this functional area:

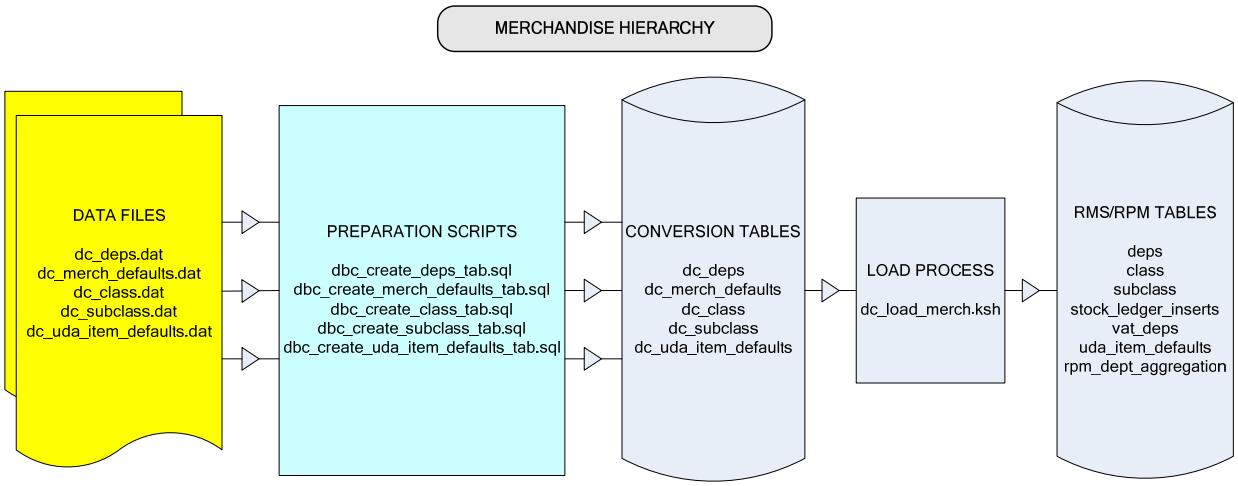
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load script. See Chapter 2, Master Script (DC_LOAD_MAIN.KSH), for details.
- Segment load script dc_load_merch.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_merch_defaults_tab.sql
  - dbc_create_deps_tab.sql
  - dbc_create_class_tab.sql
  - dbc_create_subclass_tab.sql
  - dbc_create_vat_deps.sql
  - dbc_create_uda_item_def.sql

Data Flow

The following diagram shows the 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 External Oracle 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_wh_org.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.

External Oracle Table Definition

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

Department—DC_DEPS Table

File name: **DC_DEPS.DAT**

Table create SQL script: **DBC_CREATE_DEPS_TAB.SQL**

External Oracle table created: **DC_DEPS**

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

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

- 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					EXTERNAL ORACLE 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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
MARKUP_C ALC_ 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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_AVG_ COUNTER	NUMBER(5)
AVG_ TOLER ANCE_ 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
VAT_ REGION	Integer	4	Y*	Unique identifier of VAT region. VAT region is determined by the VAT authority. This value should tie to a value from the DC_VAT_REGION.DAT file. * This field is required when VAT is turned on in RMS.	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. * This field is required when VAT is turned on in RMS.	VAT_TYPE	VARCHAR2(1)
VAT_CODE	Alpha-numeric	6	Y*	Unique identifier of value added tax code. Valid values are: S - Standard C- Composite Z - Zero E - Exemprt This value should correspond to a value from the dc_vat_codes.dat file. * This field is required when VAT is turned on in RMS.	VAT_CODE	VARCHAR2(6)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
WORK SHEET_ 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)
HISTORI CAL_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)
CLEAR ANCE_ 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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**

Table create SQL script: **DBC_CREATE_MERCH_DEFAULTS_TAB.SQL**

External Oracle table created: **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					EXTERNAL ORACLE 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 will be 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

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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_MRK UP_PCT	NUMBER
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**

Table create SQL script: **DBC_CREATE_CLASS_TAB.SQL**

External Oracle table created: **DC_CLASS**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)
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**

Table create SQL script: **DBC_CREATE_SUBCLASS_TAB.SQL**

External Oracle table created: **DC_SUBCLASS**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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**

Table create SQL script: **DBC_CREATE_VAT_DEPS.SQL**

External Oracle table created: **DC_VAT_DEPS**

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

- 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.
- The SQL script is called by the dc_load_merch.ksh and will create the DC_VAT_DEPS oracle external table if vat_ind on system_options is Y and default_tax_type is not GTAX (SVAT is used).

FILE FORMAT					**EXTERNAL ORACLE 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)

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**

Table create SQL script: **DBC_CREATE_UDA_ITEM_DEF.SQL**

External Oracle table created: **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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**Description**	**Field Name**	**Data Type**
UDA_ID	Integer	5	Y	Unique identifier of the user - defined attribute (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. No new records within a sequence group can be added to the RMS table through the scripts.

DC_LOAD_MERCH.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create the external Oracle tables.
- It calls the load data script to insert data from the external Oracle tables to the RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_merch.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_DEPS

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

If the DC_MERCH_DEFAULTS.DEFAULT_CLASS_IND is Y, only one class and subclass are inserted for each department. The DC_DEPS and DC_MERCH_DEFAULT tables are used to insert into the RMS CLASS and SUBCLASS tables. The ID and name fields are the same as department for all classes and subclasses.

The function returns a Boolean 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.

Field Name (RMS Table)	Default Value	Comments
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 files to load: dc_merch_defaults.dat, dc_deps.dat

LOAD_CLASS

This function contains a PL/SQL block that selects from the DC_CLASS external 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 dc_load_merch.ksh script determines whether to run this function when the dc_class.dat file does not exist. 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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 files to load: dc_merch_defaults.dat, dc_class.dat

LOAD _SUBCLASS

This function contains a PL/SQL block that selects from the DC_SUBCLASS external 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 dc_load_merch.ksh script will determine whether to run/not run this function. 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.

The function returns a Boolean value.

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

LOAD_STOCK_LEDGER_INS

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.

The function returns a Boolean value.

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

LOAD_VAT_DEPS

This function selects data from the DC_VAT_DEPS table and inserts the records into RMS VAT_DEPS table, if the system options vat_ind is equal to Y and default tax type is not GTAX (SVAT is used). All the columns from the external oracle table defined above will map directly to the RMS table.

The function returns a Boolean value.

Required file to load: dc_vat_deps.dat

LOAD_UDA_ITEM_DEF

This function contains a PL/SQL block that selects from the DC_UDA_ITEM_ DEFAULTS external table and inserts the data to the RMS UDA_ITEM_DEFAULTS table. All the columns from the external Oracle 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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.
HIERARCHY_VALUE	1,2 or 3	If subclass is not NULL then 3; if class is not NULL then 2; if dept is not NULL then 1.
REQUIRED_IND	N	Value defaults to N if the file value is empty.

Required file to load: dc_uda_item_defaults.dat

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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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

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.

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

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:

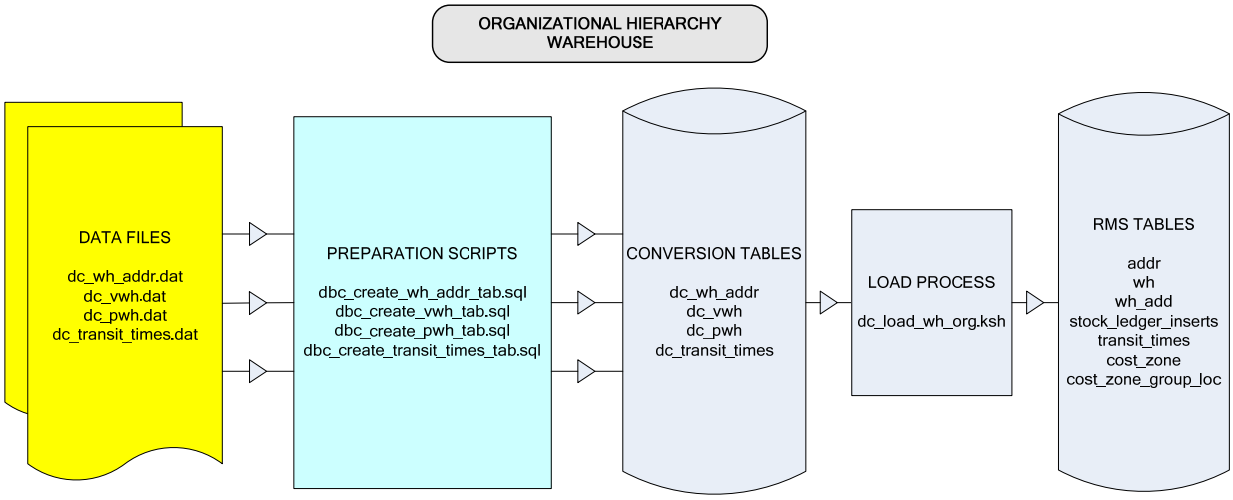
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_wh_org.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_wh_addr_tab.sql
  - dbc_create_vwh_tab.sql
  - dbc_create_pwh_tab.sql
  - dbc_create_transit_times_tab.sql

Data Flow

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



File Format and External Oracle 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 created in the order listed.

The dc_wh_org.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.

External Oracle Table Definition

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

DC_WH_ADDR Table

File name: **DC_WH_ADDR.DAT**

Table create SQL script: **DBC_CREATE_WH_ADDR_TAB.SQL**

External Oracle table created: **DC_WH_ADDR**

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

- Ensure that ADDR.KEY_VALUE_1 is a valid WH.WH. If SYSTEM_OPTION.MULTICHANNEL_IND = Y, ensure that WH.STOCKHOLDING_IND = N.
- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**Description**	**Field Name**	**Data Type**
WARE HOUSE_ 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					External Oracle Table Definition	
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_3	Alpha-numeric	240	N	Second line of the address of this warehouse and address type.	ADD_2	VARCHAR2(240)
ADDR_ LINE_2	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)

DC_PWH Table

File name: **DC_PWH.DAT**

Table create SQL script: **DBC_CREATE_PWH_TAB.SQL**

External Oracle table created: **DC_PWH**

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

- 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 (move to after running storeadd.pc).
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y and WH.STOCKHOLDING_IND = Y, ensure that WH.CHANNEL_ID is a valid CHANNELS.CHANNEL_ID.
- Ensure that WH.VAT_REGION is a valid VAT_REGION.VAT_REGION if SYSTEM_OPTIONS.VAT_IND = Y and WH.STOCKHOLIDNG_IND = Y.
- Ensure that WH.CURRENCY_CODE is a valid CURRENCIES.CURRENCY_CODE.
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y and WH.STOCKHOLDING_IND = N, ensure that WH.PRIMARY_VWH is a valid WH.WH with WH.STOCKHOLDING_IND = Y.
- 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 SYSTEM_OPTIONS.INTERCOMPANY_TRANSFER_IND = Y and WH.STOCKHOLIDNG_IND = Y.
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, capture counts from WH where WH.STOCKHOLDING_IND = N and compare to flat file DC_PWH.DAT to ensure that all rows are loaded.
- If SYSTEM_OPTION.MULTICHANNEL_IND = N, capture counts from WH and compare to flat file DC_PWH.DAT to ensure that all rows are loaded.

File Format					**External Oracle Table Definition**	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

File Format					External Oracle Table Definition	
WAREHOUSE_NAME	Alpha-numeric	150	Y	Name of the warehouse being defined.	WH_NAME	VARCHAR2(150)
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 will be 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_WH_IND	VARCHAR2(1)

File Format					External Oracle Table Definition	
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)
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). 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)

File Format					External Oracle Table Definition	
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)
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 _SECON DARY	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)

File Format					External Oracle Table Definition	
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)
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.

Table create SQL script: **DBC_CREATE_VWH_TAB.SQL**

External Oracle table created: **DC_VWH**

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

- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, ensure that WH.PHYSICAL_WH is a valid WH.WH with WH.STOCKHOLDING_IND = N.
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, capture counts from WH where WH.STOCKHOLDING_IND = Y and compare to flat file DC_VWH.DAT to ensure that all rows are loaded.

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)
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 stock will be 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					External Oracle Table Definition	
PROTECTED_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					External Oracle Table Definition	
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					External Oracle Table Definition	
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 identifer for the Oracle Organizational ID.	ORG_UNIT_ID	NUMBER(15)

DC_TRANSIT_TIMES Table

File name: **DC_TRANSIT_TIMES.DAT**

Table create SQL script: **DBC_CREATE_TRANSIT_TIMES_TAB.SQL**

External Oracle table created: **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).
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, ensure that TRANSIT_TIMES.DESTINATION is a valid WH.WH, where WH.STOCKHOLDING_IND = N when TRANSIT_TIMES.DESTINATION_TYPE = WH.
- If SYSTEM_OPTION.MULTICHANNEL_IND = N, ensure that TRANSIT_TIMES.DESTINATION is a valid WH.WH when TRANSIT_TIMES.DESTINATION_TYPE = WH.
- Capture the count from TRANSIT_TIMES and compare to flat file DC_TRANSIT_TIMES.DAT to ensure that all rows are loaded.

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)
DESTINA TION	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)
DESTINA TION_ 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					External Oracle Table Definition	
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)

DC_LOAD_WH_ORG.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_WH_ADDR

This function contains a PL/SQL block that selects from the DC_WH_ADDR external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_WH_ADDR to ADDR Column Defaults

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

Required file to load: dc_wh_addr.dat

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 external 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 only when SYSTEM_OPTIONS. MULTICHANNEL_IND = Y. 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. If SYSTEM_OPTIONS. MULTICHANNEL_IND = Y, it inserts the virtual warehouse number. 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_PWH to WH, WH_ADD, STOCK_LEDGER_INSERTS Column Defaults

Column Name (RMS Table)	Default Value	Comments
RESTRICTED_IND	N	
PROTECTED_IND	N	
BREAK_PACK_IND	Y	If NULL
REDIST_WH_IND	Y	If NULL
FORECAST_WH_IND	Y	If NULL
REPL_IND		If multichannel = Y, then override file value with N; otherwise, default to Y.
IB_IND	N	
STOCKHOLDING_IND		N if multi-channel, Y if not multi-channel
AUTO_IB_CLEAR	N	
FINISHER_IND	N	This can only be Yes for virtual warehouses in a multi-channel environment, so always set it to N.
PHYSICAL_WH		WAREHOUSE_ID
ORG_ENTITY_TYPE	R	If multichannel = Y, then override file value with N; otherwise, default to Y.

DC_VWH to WH, WH_ADD, STOCK_LEDGER_INSERTS Column Defaults

Column Name (RMS Table)	Default Value	Comments
STOCKHOLDING_IND	Y	
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_IND	N	WAREHOUSE_ID
VAT_REGION		From physical warehouse
CURRENCY_CODE		From physical warehouse
ORG_HIER_TYPE		From physical warehouse
ORG_HIER_VALUE		From physical warehouse
DELIVERY_POLICY		From physical warehouse
EMAIL		From physical warehouse
DUNS_NUMBER		From physical warehouse
DUNS_LOC		From physical warehouse
INBOUND_HANDLING_DAYS		From physical warehouse
BREAK_PACK_IND		From physical warehouse
REDIST_WH_IND		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.

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

LOAD_TRANSIT_TIMES

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

Required file to load: dc_transit_times.dat

INSERT_COST_ZONE_LOCS

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 external 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 external 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, if SYSTEM_OPTIONS multichannel_ind is set to Y. 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 (if multi-channel is active)

Store

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

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

The following programs are included in this functional area:

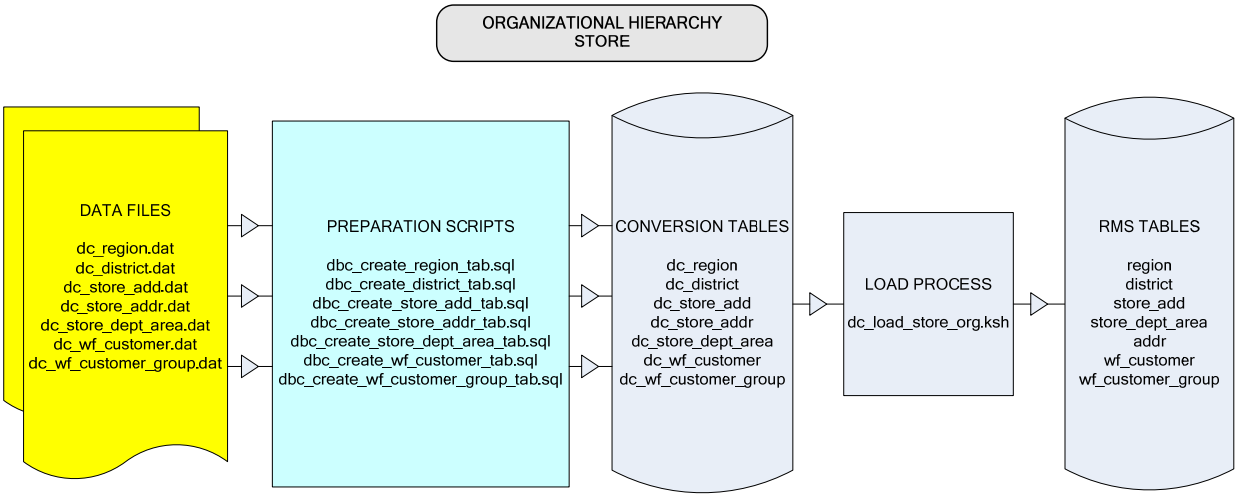
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_store_org.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_region_tab.sql
  - dbc_create_district_tab.sql
  - dbc_create_store_add_tab.sql
  - dbc_create_store_addr_tab.sql
  - dbc_create_store_dept_area.sql
  - dbc_create_wf_customer_tab.sql
  - dbc_create_wf_customer_group_tab.sql

Data Flow

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



File Format and External Oracle 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_store_org.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.

External Oracle Table Definition

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

DC_REGION Table

File name: **DC_REGION.DAT**

Table create SQL script: **DBC_CREATE_REGION_TAB.SQL**

External Oracle table created: **DC_REGION**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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**

Table create SQL script: **DBC_CREATE_DISTRICT_TAB.SQL**

External Oracle table created: **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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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**

Table create SQL script: **DBC_CREATE_STORE_ADDR_TAB.SQL**

External Oracle table created: **DC_STORE_ADDR**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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.	ADDR_TYPE	VARCHAR2(2)

File Format					External Oracle Table Definition	
				The required address types for a store are definable in the RMS ADD_TYPE_MODULE table, where MODULE = ST for company stores and WFST for wholesale/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)

DC_STORE_ADD Table

File name: **DC_STORE_ADD.DAT**

Table create SQL script: **DBC_CREATE_STORE_ADD_TAB.SQL**

External Oracle table created: **DC_STORE_ADD**

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

- Ensure that STORE_ADD.STORE is unique and does not exist on STORE.
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, ensure that STORE_ADD.CHANNEL_ID is a valid CHANNELS.CHANNEL_ID.
- Ensure that STORE_ADD.VAT_REGION is a valid VAT_REGION.VAT_REGION, if SYSTEM_OPTIONS.VAT_IND = Y.
- 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.TSF_ENTITY_ID is a valid TSF_ENTITYT.TSF_ENTITY_ID, if SYSTEM_OPTIONS.INTERCOMPANY_TRANSFER_IND = Y.
- Ensure that STORE_ADD.STORE_FORMAT (if not NULL) is a valid STORE_FORMAT.STORE_FORMAT.

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

File Format					External Oracle Table Definition	
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 if the multi-channel indicator is Y or store_type = W or F (wholesale/ franchise). Override to Y when multi-channel is set to 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)
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)

File Format					External Oracle Table Definition	
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
ACQUIRED_DATE	Alpha-numeric	9	N	Date the store was acquired. Date format is DDMONYYYY.	ACQUIRED_DATE	DATE
REMODEL_DATE	Alpha-numeric	9	N	Last date the store was remodeled. Date format is DDMONYYYY.	REMODEL_DATE	DATE
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)

File Format					External Oracle Table Definition	
VAT_REGION	Integer	4	N	Number of the value added tax region in which this store is located. Required if SYSTEM_OPTIONS.VAT_IND = Y, even 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)
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)

File Format					External Oracle Table Definition	
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)
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 W – Wholesale F - Franchise	STORE_TYPE	VARCHAR2(6)

File Format					External Oracle Table Definition	
WF_CUSTOMER_ID	Integer	10	N	Unique ID of the wholesale or 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)

DC_STORE_DEPT_AREA Table

File name: **DC_STORE_DEPT_AREA.DAT**

Table create SQL script: **DBC_CREATE_STORE_DEPT_AREA_TAB.SQL**

External Oracle table created: **DC_STORE_DEPT_AREA**

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

- Ensure that STORE_DEPT_AREA.STORE is a valid STORE_ADD.STORE or a valid STORE.STORE.
- Ensure that STORE_DEPT_AREA.DEPT is a valid DEPS.DEPT.
- Capture the count from STORE_DEPT_AREA and compare to flat file DC_STORE_DEPT_AREA.DAT to ensure that all rows are loaded.

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**Description**	**Field Name**	**Data Type**
STORE	Integer	10	Y	Unique ID of the store.	STORE	NUMBER(10)
MERCH_HIER_4	Integer	4	Y	Fourth level of the merchandise hierarchy, referred to as department in the base configuration. This field must contain a valid value from the RMS DEPS table.	DEPT	NUMBER(4)
EFFECTIVE_DATE	Date	7	Y	Date on which the area is effective for the store and hierarchy. Date format is DDMONYYYY (for example, 02JAN2011).	EFFECTIVE_DATE	DATE(7)
AREA	Number	12,4	Y	Area in the store used by the hierarchy value (department).	AREA	NUMBER(12,4)

DC_WF_CUSTOMER Table

File name: **DC_WF_CUSTOMER.DAT**

Table create SQL script: **DBC_CREATE_WF_CUSTOMER_TAB.SQL**

External Oracle table created: **DC_WF_CUSTOMER**

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**Description**	**Field Name**	**Data Type**
WF_ CUSTOMER_ ID	Integer	10	Y	Unique ID of the wholesale or franchise store.	WF_CUSTOMER _ID	NUMBER(10)
WF_ CUSTOMER_ NAME	Alpha-numeric	120	Y	Name of the wholesale or franchise store.	WF_CUSTOMER _NAME	VARCHAR2(120)
WF_ CUSTOMER_ TYPE	Alpha-numeric	6	Y	Type of customer store. Valid values are: W – Wholesale F – Franchise	WF_CUSTOMER _TYPE	VARCHAR2(6)
CREDIT_IND	Alpha-numeric	1	Y	Indicates whether the wholesale or 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 wholesale or franchise store.	WF_CUSTOMER _GROUP_ID	NUMBER(10)

DC_WF_CUSTOMER_GROUP Table

File name: **DC_WF_CUSTOMER_GROUP.DAT**

Table create SQL script: **DBC_CREATE_WF_CUSTOMER_GROUP_TAB.SQL**

External Oracle table created: **DC_WF_CUSTOMER_GROUP**

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

DC_LOAD_STORE_ORG.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_store_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_ REGION

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

Required file to load: dc_region.dat

LOAD_ DISTRICT

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

Required file to load: dc_district.dat

LOAD_STORE_ADDRESS

This function contains a PL/SQL block that selects from the DC_STORE_ADDR external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_STORE_ADDR to ADDR Column Defaults

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

Required file to load: dc_store_addr.dat

LOAD_ STORE_ADD

This function contains a PL/SQL block that selects from the DC_STORE_ADD external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_STORE_ADD to STORE_ADD Column Defaults

Column Name (RMS Table)	Default Value	Comments
STOCKHOLDING_IND	Y	Defaults to N if STORE_TYPE = W or F. Otherwise, it defaults to Y when multi-channel is off. When multi-channel is on, defaults to Y when the value is NULL.
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
STORE_TYPE	C	If NULL.
AUTO_APPROVE_ORDERS_IND	N	If NULL.

Required file to load: dc_store_add.dat

LOAD_STORE_DEPT_AREA_TEMP()

This function contains a PL/SQL block that selects from DC_STORE_DEPT_AREA external table and inserts the data to the RMS STORE_DEPT_AREA table.

The Storeadd batch takes care of inserting the data from STORE_DEPT_AREA_TEMP table to RMS STORE_DEPT_AREA table.

Required file to load: dc_store_dept_area.dat

LOAD_WF_CUSTOMER

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

Required file to load: dc_wf_customer.dat

LOAD_WF_CUSTOMER_GROUP

This function selects all columns from the DC_WF_CUSTOMER_GROUP external table and inserts data into the WF_CUSTOMER_GROUP table. The DC_WF_CUSTOMER_GROUP external table maps exactly to the RMS WF_CUSTOMER_GROUP table.

Required file to load: dc_wf_customer_group.dat

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

STOREADD.PC Batch

Run the storeadd.pc batch program at the end, to load store data from the RMS STORE_ADD table into RMS. When a store record is added to the RMS STORE_ADD table, the store data is accessible in the system only after the storeadd.pc batch program is run. The batch program loops through each record in the STORE_ADD table and performs all the necessary inserts into the different RMS tables. This program adds all information necessary for a new store to function properly. For details about storeadd.pc, please refer to the Oracle Retail Merchandising System Operations Guide.

WHADD.PC Batch

Run the whadd.pc batch program at the end, to load data from the RMS WH_ADD table into RMS. This batch program inserts pricing/zone information for new warehouses, virtual warehouses, and internal finishers. It reads from the WH_ADD table and inserts into the PRICE_ZONE and PRICE_ZONE_GROUP_STORE tables for each retrieved record. Successfully processed records are deleted from the WH_ADD table. For more information about the whadd.pc batch program, refer to the Oracle Retail Merchandising System Operations Guide.

6

Suppliers

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

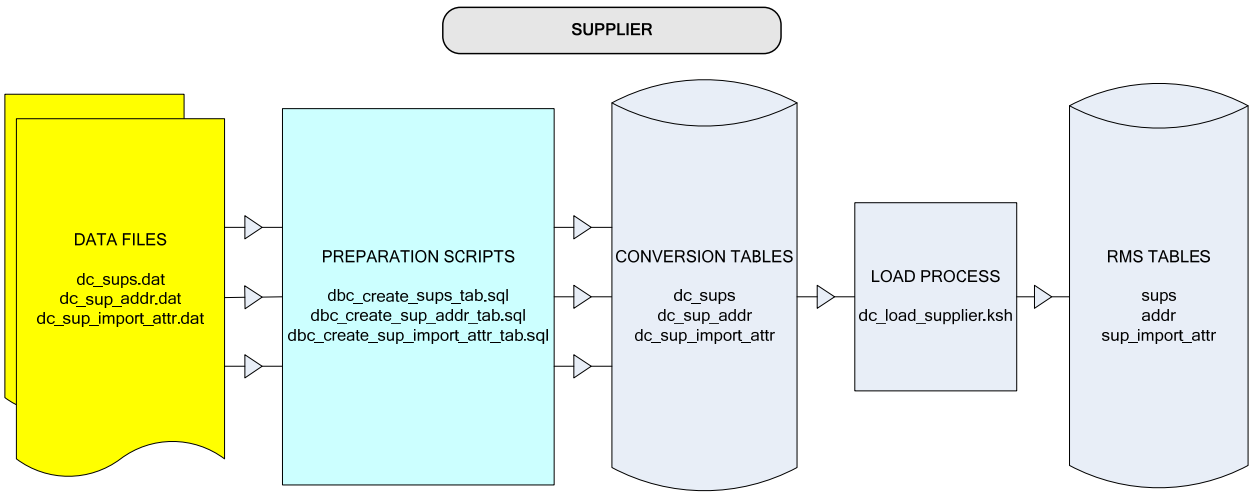
- SUPS
- ADDR (for supplier addresses)
- SUP_IMPORT_ATTR
- PARTNER

The following programs are included in the Suppliers functional area:

- Main wrapper script dc_load_main.ksh
- This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_supplier.ksh
- This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_sups_tab.sql
  - dbc_create_sup_addr_tab.sql
  - dbc_create_sup_import_attr_tab.sql
  - dbc_create_partner_tab.sql

Data Flow

The following diagram shows the 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 External Oracle 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.

External Oracle Table Definition

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

Suppliers—DC_SUPS Table

File name: **DC_SUPS.DAT**

Table create SQL script: **DBC_CREATE_SUPS_TAB.SQL**

External Oracle table created: **DC_SUPS**

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

- Ensure that SUPS.SUPPLIER is unique.
- If SYSTEM_OPTION.MULTICHANNEL_IND = Y, ensure that SUPS.EDI_CHANNEL_ID (if not NULL) is a valid CHANNELS.CHANNEL_ID.
- 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.
- Ensure that SUPS.VAT_REGION is a valid VAT_REGION.VAT_REGION if SYSTEM_OPTIONS.VAT_IND = Y.
- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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					External Oracle Table Definition	
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					External Oracle Table Definition	
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)
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)

File Format					External Oracle Table Definition	
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)
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)

File Format					External Oracle Table Definition	
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.	SETTLEMENT_ CODE	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)
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)

File Format					External Oracle Table Definition	
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_ITE M_LEAD_TIME	NUMBER(4)
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)
BRACKET_ COSTING_IND	Alpha-numeric	1	Y	Indicates whether the supplier accepts returns. Valid values are Y and N.	BRACKET_ COSTING_IND	VARCHAR2(1)
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)

File Format					External Oracle Table Definition	
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**

Table create SQL script: **DBC_CREATE_SUP_ADDR_TAB.SQL**

External Oracle table created: **DC_SUP_ADDR**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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	ADDR_TYPE	VARCHAR2(2)

File Format					External Oracle Table Definition	
				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.		
PRIMARY_ ADDR_IND	Alpha-numeric	1	Y	Indicates whether the address is the primary address for this address type.	PRIMARY_ ADDR_IND	VARCHAR2(1)
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)

Supplier Import Attributes—DC_SUP_IMPORT_ATTR Table

File name: **DC_SUP_IMPORT_ATTR.DAT**

Table create SQL script: **DBC_CREATE_SUP_IMPORT_ATTR_TAB.SQL**

External Oracle table created: **DC_SUP_IMPORT_ATTR**

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

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

File Format					External Oracle Table Definition	
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)
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)

File Format					External Oracle Table Definition	
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)
FACTORY	Alphanumeric	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)

DC_LOAD_SUPPLIER.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS table.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_supplier.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_SUPPLIER

This function selects from the DC_ SUPS external table and inserts the data to the RMS SUPS table. All the columns from the external Oracle 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.

The function returns a Boolean value.

DC_SUPS to SUPS Column Defaults

Field Name (RMS Table)	Default Value	Comments
SUP_STATUS	A	
AUTO_APPR_DBT_MEMO_IND	Y	If NULL in external table
PREPAY_INVC_IND	Y	
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	
DBT_MEMO_CODE	Y	
INVC_PAY_LOC	C	
INVC_RECEIVE_LOC	C	
ADDINVC_GROSS_NET	G	
VMI_ORDER_STATUS	A	If NULL in external table

Required file to load: dc_sups.dat

LOAD_SUP_ADDR

This function selects from the DC_ SUP_ADDR external table and inserts the data to the RMS ADDR table. All the columns from the external Oracle 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.

The function returns a Boolean value.

DC_SUP_ADDRESS to ADDR Column Defaults

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

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

LOAD_ SUP_IMPORT_ATTR

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

The function returns a Boolean value.

Required file to load: dc_sup_import_attr.dat

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.

Partner

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:

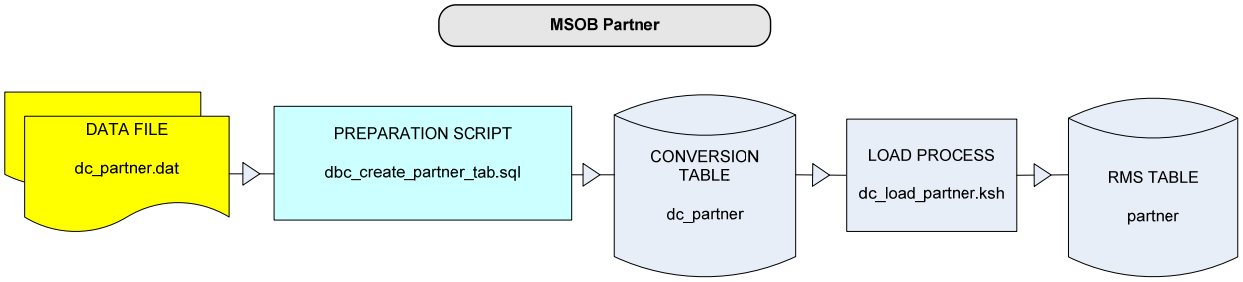
- Main wrapper dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.
- Segment load script dc_load_partner.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts: dbc_create_partner_tab.sql

Data Flow

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



File Format and External Oracle 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_partnerwh_org.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.

External Oracle Table Definition

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

DC_PARTNER Table

File name: **DC_PARTNER.DAT**

Table create SQL script: **DBC_CREATE_PARTNER_TAB.SQL**

External Oracle table created: **DC_PARTNER**

Suggested post-loading validation (sequence after dc_load_partner.ksh:

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**Description**	**Field Name**	**Data Type**
PARTNER_TYPE	Alpha-numeric	6	Y	Specifies type of partner, such as Bank (BK).	PARTNER_TYPE	VARCHAR2(6)

File Format					External Oracle Table Definition	
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	Numeric	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	Contact phone number.	CONTACT_ PHONE	VARCHAR2(20)
CONTACT_ FAX	Alpha-numeric	20	N	Contact fax number.	CONTACT_ FAX	VARCHAR2(20)
CONTACT_ TELEX	Alpha-numeric	20	N	Contact telex number.	CONTACT_ TELEX	VARCHAR2(20)
CONTACT_ EMAIL	Alpha-numeric	100	N	Contact email address.	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 the partner is assigned.	PRINCIPLE_ COUNTRY_ ID	VARCHAR2(3)
LINE_OF_ CREDIT	Numeric	16	N	The line of credit the company has at the bank (in partner's currency).	LINE_OF_ CREDIT	NUMBER(16)
OUTSTAND_ CREDIT	Numeric	16	N	Total amount of credit that the company has used or charged against in the partner's currency.	OUTSTAND_ CREDIT	NUMBER(16)
OPEN_ CREDIT	Numeric	16	N	Total amount that the company can still charge against in the partner's currency.	OPEN_ CREDIT	NUMBER(16)
YTD_ CREDIT	Numeric	16	N	Total amount of credit the company has used this year to date.	YTD_ CREDIT	NUMBER(16)
YTD_ DRAW DOWNS	Numeric	16	N	The year-to-date payments the bank has made on behalf of the company.	YTD_ DRAW DOWNS	NUMBER(16)
TAX_ID	Alpha-numeric	18	N	Unique tax identification number.	TAX_ID	VARCHAR2(18)

File Format					External Oracle Table Definition	
TERMS	Alpha-numeric	15	Y	Payment terms for the 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.	INVC_PAY_ LOC	VARCHAR2(6)
INVC_ RECEIVE_ LOC	Alpha-numeric	6	N	Indicate 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)
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	Comments pertaining to the partner.	COMMENT_ DESC	VARCHAR2 (2000)
TSF_ENTITY_ ID	Numeric	10	N	ID of transfer entity with which external finisher is associated.	TSF_ENTITY_ ID	NUMBER(10)
VAT_ REGION	Numeric	4	N	VAT region with which partner is associated.	VAT_ REGION	NUMBER(4)
ORG_UNIT_ ID	Numeric	15	N	Organization unit ID.	ORG_UNIT_ ID	NUMBER(15)
PARTNER_ NAME_ SECONDARY	Alpha-Numeric	240	N	This wil 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 first leg of the transfer is shipped. Valid values are Y or N.	AUTO_ RECEIVE_ IND	VARCHAR2(1)

DC_LOAD_PARTNER.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh. It serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just to load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_wh.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

LOAD_PARTNER

This function contains a PL/SQL block that selects from the DC_PARTNER external table and inserts the data to the RMS PARTNER table. All the columns from the external Oracle 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

The function returns a Boolean value.

Required file to load: dc_partner.dat

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

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:

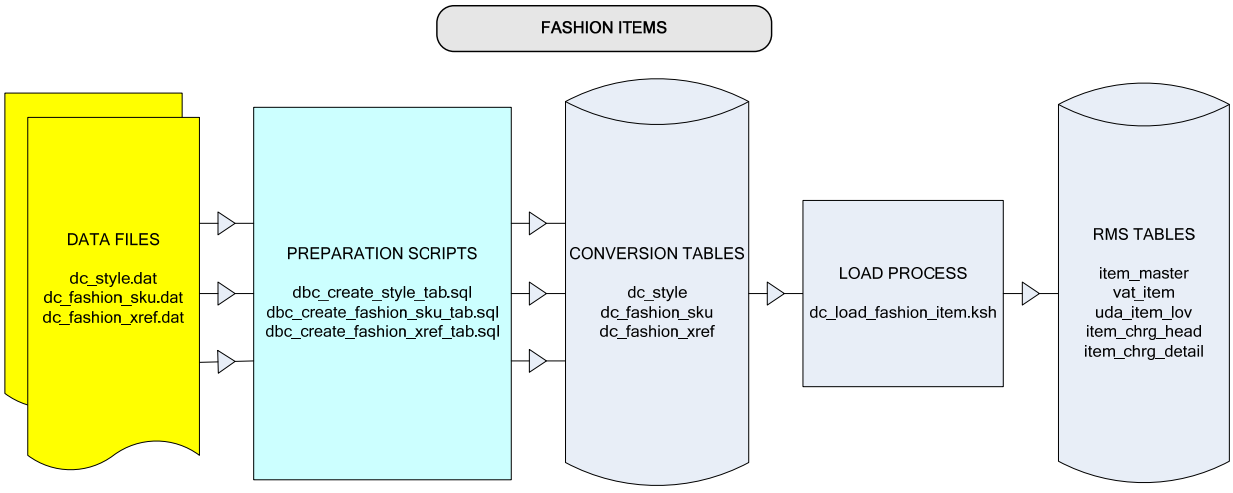
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load script. Refer to Chapter 2 for details.
- Segment load script dc_load_fashion_item.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_style_tab.sql
  - dbc_create_fashion_sku_tab.sql
  - dbc_create_fashion_xref_tab.sql

Data Flow

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



File Format and External Oracle 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_fashion_item.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.

External Oracle Table Definition

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

DC_STYLE Table

File name: **DC_STYLE.DAT**

Table create SQL script: **DBC_CREATE_STYLE_TAB.SQL**

External Oracle table created: **DC_STYLE**

Suggested post-loading validation (sequence after dc_load_fashion_item.ksh:

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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_ SECON DARY	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)

File Format					External Oracle Table Definition	
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)
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)
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_ AGGRE GATE	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_ AGGRE GATE	VARCHAR2(1)

File Format					External Oracle Table Definition	
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_ AGGRE GATE	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_ AGGRE GATE	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_ AGGRE GATE	VARCHAR2(1)
STYLE_ COMMENTS	Alpha-numeric	2000	N	Comments associated with the style.	STYLE_ COMMENTS	VARCHAR2 (2000)

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**

Table create SQL script: **DBC_CREATE_FASHION_SKU_TAB.SQL**

External Oracle table created: **DC_FASHION_SKU**

Suggested post-loading validation (sequence after dc_load_fashion_item.ksh:

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)
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)

File Format					External Oracle Table Definition	
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	MERCHAN DISE_IND	VARCHAR2(1)
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)

File Format					External Oracle Table Definition	
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)

DC_FASHION_XREF Table

File name: **DC_FASHION_XREF.DAT**

Table create SQL script: **DBC_CREATE_FASHION_XREF_TAB.SQL**

External Oracle table created: **DC_FASHION_XREF**

Suggested post-loading validation (sequence after dc_load_fashion_item.ksh:

- 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.RETAIL_ZONE_GROUP_ID is a valid PRICE_ZONE_GROUP.ZONE_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.
- Ensure that ITEM_ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = UPCT.

FILE FORMAT					**EXTERNAL ORACLE 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_SEC ONDARY	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_GRAND PARENT	VARCHAR2(25)
XREF_COM MENTS	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 etc.	ITEM_NUMBER_ TYPE	VARCHAR2(6)

DC_LOAD_FASHION_ITEM.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_fashion_item.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_ STYLE_SKU

This function contains a PL/SQL block that selects from the DC_STYLE and the DC_FASHION_SKU external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_STYLE and DC_FASHION_SKU to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	1	-
TRAN_LEVEL	2	-
SHORT_DESC	SUBSTR 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	SYSDATE	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	SYSDATE	-

Column Name (RMS Table)	Default Value	Comments
RETAIL_ZONE_GROUP_ID	Lookup in PRICE_ZONE_GROUP table	-
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	-

SKUs

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

The function returns a Boolean value.

DC_FASHION_SKU to ITEM_MASTER Column Defaults

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

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

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

INSERT_ITEM_DEFAULTS

This function inserts item defaults from the merchandise hierarchy specifications for VAT, UDAs and item charges.

Using bulk collect, this function retrieves into a PL/SQL table the ITEM, DEPT, CLASS and SUBCLASS values from DC_HARDLINES from the DC_STYLE table and from DC_STYLE joined with DC_FASHION_SKU.

If the VAT indicator is turned on in system_options and default tax type is not GTAX (SVAT is used), the function retrieves SKU information and calls the VAT_SQL.DEFAULT_VAT_ITEM to default data into RMS VAT_ITEM table.

It also retrieves style information and calls UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL, DC_DEFAULT_CHRGS. It retrieves SKU 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 files to load: dc_style.dat, dc_fashion_sku.dat

INSERT_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), retreive 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. Retreive sku information and call UDA_SQL.INSERT_DEFAULTS and ITEM_CHARGE_SQL.DEFAULT_CHRGS.

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

LOAD_XREF_ITEMS

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

Most of the columns from the external Oracle 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_FASHION_XREF and DC_FASHION_SKU to ITEM_MASTER Column Defaults

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

Required files to load: dc_style.dat, dc_fashion_sku.dat,dc_fashion_xref.dat

Hardline Items

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

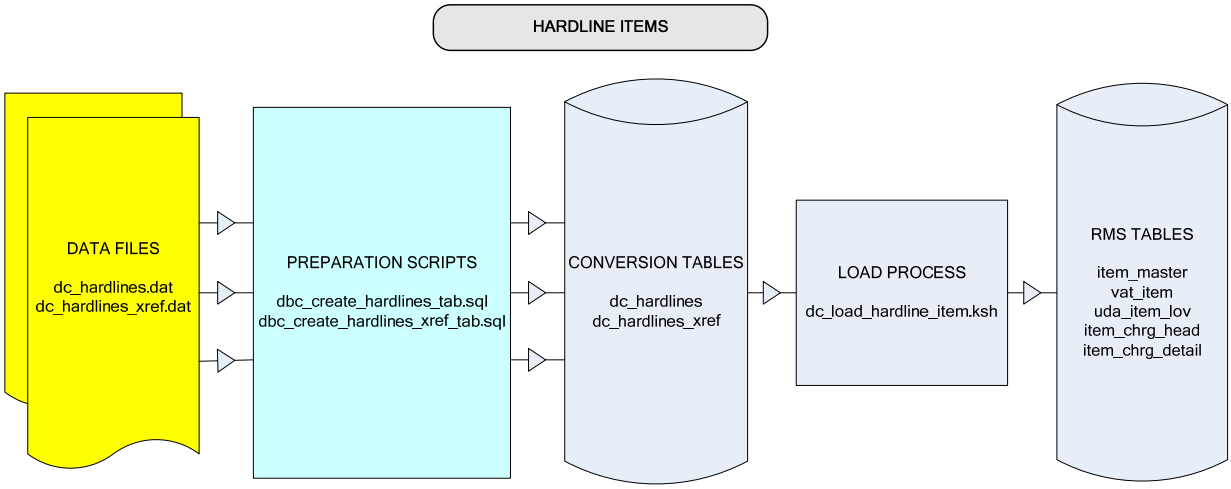
The following programs are included in this functional area.

- Main wrapper script dc_load_main.ksh
- This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.

- Segment load script dc_load_hardline_item.ksh. This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_hardlines_tab.sql
  - dbc_create_hardlines_xref_tab.sql

Data Flow

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



File Format and External Oracle 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_hardline_item.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.

External Oracle Table Definition

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

DC_HARDLINES Table

File name: **DC_HARDLINES.DAT**

Table create SQL script: **DBC_CREATE_HARDLINES_TAB.SQL**

External Oracle table created: **DC_HARDLINES**

Suggested post-loading validation (sequence after dc_load_hardline_item.ksh:

- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)
STAN DARD_ 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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**

Table create SQL script: **DBC_CREATE_HARDLINES_XREF_TAB.SQL**

External Oracle table created: **DC_HARDLINES_XREF**

Suggested post-loading validation (sequence after dc_load_hardline_item.ksh:

- 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_MASTER.RETAIL_ZONE_GROUP_ID is a valid PRICE_ZONE_GROUP.ZONE_GROUP_ID.
- Ensure that ITEM_ITEM_NUMBER_TYPE is a valid CODE_DETAIL.CODE, where CODE_DETAIL.CODE_TYPE = UPCT.

FILE FORMAT					EXTERNAL ORACLE 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)

DC_LOAD_HARDLINE_ITEM.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_hardline_item.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_HARDLINES

This function contains a PL/SQL block that selects from the DC_HARDLINES external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_HARDLINES to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	1	-
TRAN_LEVEL	1	-
SHORT_DESC	rtrim of substr b 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	sysdate	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	sysdate	-
RETAIL_ZONE_GROUP_ID	Look up price_zone_group table	-
ORDERABLE_IND	Y	-
SELLABLE_IND	Y	-
INVENTORY_IND	Y	-
MERCHANDISE_IND	Y	If NULL
FORECAST_IND	Y	If NULL
ITEM_AGGREGATE_IND	N	-

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

Required file to load: dc_hardlines.dat

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

LOAD_HARDLINES_XREF

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

Most of the columns from the external Oracle 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_HARDLINES_XREF to ITEM_MASTER Column Defaults

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

Required files to load: dc_hardlines.dat, dc_hardlines_xref.dat

Grocery Items

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

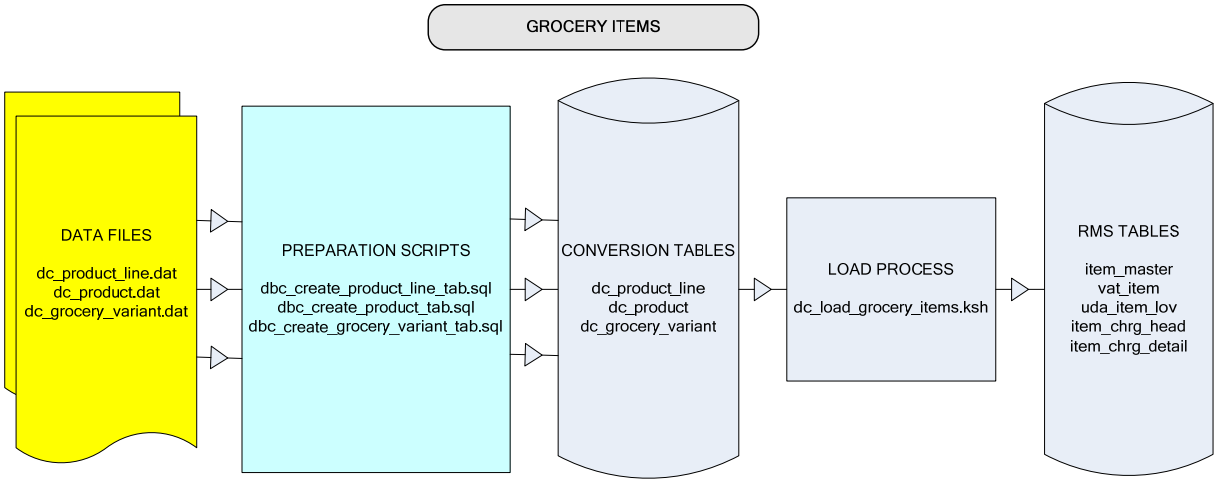
The following programs are included in this functional area:

- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.
- Segment load script dc_load_grocery_items.ksh.
- This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_product_line_tab.sql
  - dbc_create_product_tab.sql
  - dbc_create_grocery_variant_tab.sql

Data Flow

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



File Format and External Oracle 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_grocery_items.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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**

Table create SQL script: **DBC_CREATE_PRODUCT_LINE_TAB.SQL**

External Oracle table created: **DC_PRODUCT_LINE**

Suggested post-loading validation (sequence after dc_load_grocery_items.ksh:

- 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					**EXTERNAL ORACLE TABLE DEFINITION**	
Field Name	**Data Type**	**Max Length**	**Req**	**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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
PRODUCT_ LINE_ COMMENTS	Alpha-numeric	2000	N	Comments associated with the product line.	COMMENTS	VARCHAR2 (2000)

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**

Table create SQL script: **DBC_CREATE_PRODUCT_TAB.SQL**

External Oracle table created: **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					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req**	**Description**	**Field Name**	**Data Type**
PRODUCT	Alpha-numeric	25	Y	ID that identifies the product.	ITEM	VARCHAR2(25)
PRIMARY_ PRODUCT_I ND	Alpha-numeric	1	N	Not in the RMS ITEM_MASTER table, needed for defaulting product line.	PRIMARY_ PRODUCT_IND	VARCHAR2(1)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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					EXTERNAL ORACLE TABLE DEFINITION	
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					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

DC_GROCERY_VARIANT Table

File name: **DC_GROCERY_VARIANT.DAT**

Table create SQL script: **DBC_CREATE_GROCERY_VARIANT_TAB.SQL**

External Oracle table created: **DC_GROCERY_VARIANT**

Suggested post-loading validation (sequence after dc_load_grocery_items.ksh:

- 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.RETAIL_ZONE_GROUP_ID is a valid PRICE_ZONE_GROUP.ZONE_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.
- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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 product.	PRIMARY_REF_ ITEM_IND	VARCHAR2(1)
VARIANT_ COMMENTS	Alpha-numeric	2000	N	Comments associated with the variant.	COMMENTS	VARCHAR2 (2000)

DC_LOAD_GROCERY_ITEMS.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_grocery_items.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_PRODUCT_LINE

This function contains a PL/SQL block that selects from the DC_PRODUCT_LINE and DC_PRODUCT external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_PRODUCT_LINE and DC_PRODUCT to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	1	-
TRAN_LEVEL	2	-
SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	sysdate	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	sysdate	-
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, dc_product.dat

LOAD_PRODUCT

This function contains a PL/SQL block that selects from the DC_PRODUCT external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_PRODUCT_LINE and DC_PRODUCT to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	2	-
TRAN_LEVEL	2	-
SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	sysdate	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	sysdate	-
RETAIL_ZONE_GROUP_ID	Lookup from price_zone_group table	-
ORDERABLE_IND	Y	-
SELLABLE_IND	Y	-
INVENTORY_IND	Y	-
MERCHANDISE_IND	Y	If NULL
FORECAST_IND	Y	If NULL
ITEM_AGGREGATE_IND	N	-
DIFF_1_AGGREGATE_IND	N	-
DIFF_2_AGGREGATE_IND	N	-
DIFF_3_AGGREGATE_IND	N	-
DIFF_4_AGGREGATE_IND	N	-
PRIMARY_REF_ITEM_IND	N	-
CONST_DIMEN_IND	N	-
GIFT_WRAP_IND	N	-
SHIP_ALONE_IND	N	-
ITEM_XFORM_IND	N	-
PACK_IND	N	-
SIMPLE_PACK_IND	N	-
CATCH_WEIGHT_IND	N	If NULL
CONTAINS_INNER_IND	N	-
PERISHABLE_IND	N	-

Required file to load: dc_product_line.dat, dc_product.dat

LOAD_GROCERY_VARIANT

This function contains a PL/SQL block that selects from the DC_GROCERY_VARIANT external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_GROCERY_VARIANT and DC_HARDLINES to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_LEVEL	3	-
TRAN_LEVEL	2	-
SHORT_DESC	RTRIM/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
PRIMARY_REF_ITEM_IND	N	If NULL
PERISHABLE_IND	N	If NULL

Required files to load: dc_product_line.dat, dc_product.dat, dc_grocery_variant.dat

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:

- 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.
- 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.
- 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

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:

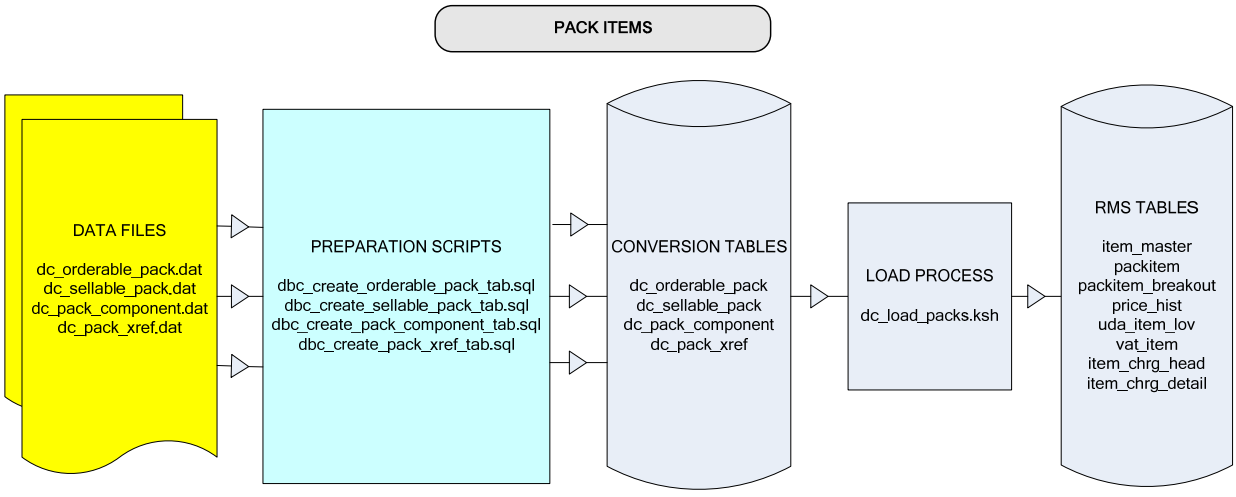
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_packs.ksh

  This wrapper calls the external Oracle table create scripts listed below.
- External Oracle table create scripts:
  - dbc_create_orderable_pack_tab.sql
  - dbc_create_sellable_pack_tab.sql
  - dbc_create_pack_component_tab.sql
  - dbc_create_pack_xref_tab.sql

Data Flow

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



File Format and External Oracle 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_packs.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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.

Table create SQL script: **DBC_CREATE_ORDERABLE_PACK_TAB.SQL**

External Oracle table created: **DC_ORDERABLE_PACK**

Suggested post-loading validation (sequence after dc_load_packs.ksh:

- 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					**EXTERNAL ORACLE 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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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 down stream 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)

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.

Table create SQL script: **DBC_CREATE_SELLABLE_PACK_TAB.SQL**

External Oracle table created: **DC_SELLABLE_PACK**

Suggested post-loading validation (sequence after dc_load_packs.ksh:

- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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.	SUB CLASS	NUMBER(4)
PACK_ DESCRIP TION	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_ SECON DARY_ 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

DC_PACK_COMPONENT Table

File name: **DC_PACK_COMPONENT.DAT**

Table create SQL script: **DBC_CREATE_PACK_COMPONENT_TAB.SQL**

External Oracle table created: **DC_PACK_COMPONENT**

Suggested post-loading validation (sequence after dc_load_packs.ksh:

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_PACK_XREF_TAB.SQL**

External Oracle table created: **DC_PACK_XREF**

Suggested post-loading validation (sequence after dc_load_packs.ksh:

- 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					EXTERNAL ORACLE 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)
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_ COM MENTS	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)

DC_LOAD_PACKS.KSH Segment Wrapper / Load Script

This ksh script is called by the Master Script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create the external Oracle tables.
- It calls the load data script to insert data from the external Oracle tables to the RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c) this script loads the data.

The dc_load_packs.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

DC_PACK_COMPONENT Table

File name: **DC_PACK_COMPONENT.DAT**

Table create SQL script: **DBC_ CREATE_PACK_COMPONENT_TAB.SQL**

External Oracle table created: **DC_PACK_ COMPONENT**

FILE FORMAT					EXTERNAL ORACLE 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	Numeric	12,4	Y	Quantity of the Item within the pack	PACK_ITEM_QTY	NUMBER(12,4)

LOAD_ORDERABLE_PACK

This function contains a PL/SQL block that selects from the DC_ORDERABLE_PACK external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_ORDERABLE_PACK to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	1	-
TRAN_LEVEL	1	-
SHORT_DESC	substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	sysdate	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	sysdate	-
MFG_REC_RETAIL		Ensure that sellable_ind = Y, otherwise NULL
RETAIL_ZONE_GROUP_ID	Lookup from price_zone_group table	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	-
STORE_ORD_MULT		If NULL, E
ORDERABLE_IND	Y	-
INVENTORY_IND	Y	-
PACK_TYPE		Ensure V if simple pack
ORDER_AS_TYPE		Ensure pack_type = B and simple_pack_ind = N, otherwise null.
ITEM_AGGREGATE_IND	N	-
DIFF_1_AGGREGATE_IND	N	-
DIFF_2_AGGREGATE_IND	N	-

Column Name (RMS Table)	Default Value	Comments
DIFF_3_AGGREGATE_IND	N	-
DIFF_4_AGGREGATE_IND	N	-
PRIMARY_REF_ITEM_IND	N	-
CONST_DIMEN_IND	N	-
GIFT_WRAP_IND	N	-
SHIP_ALONE_IND	N	-
ITEM_XFORM_IND	N	-
PACK_IND	Y	-
MERCHANDISE_IND	Y	-
FORECAST_IND	N	-
CONTAINS_INNER_IND	N	-
NOTIONAL_PACK_IND	N	If NULL in External Table.
SOH_INQUIRY_AT_PACK_IND	N	If NULL in External Table.
PERISHABLE_IND	N	-

Required file to load: dc_orderable_pack.dat

LOAD_SELLABLE_PACK

This function contains a PL/SQL block that selects from the DC_SELLABLE_PACK external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_SELLABLE_PACK to ITEM_MASTER Column Defaults

Column Name (RMS Table)	Default Value	Comments
ITEM_NUMBER_TYPE	ITEM	-
ITEM_LEVEL	1	-
TRAN_LEVEL	1	-
SHORT_DESC	Rtrim/substrb 120 characters from ITEM_DESC	If NULL
DESC_UP	Upper ITEM_DESC	-
STATUS	A	-
CREATE_DATETIME	sysdate	-
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	sysdate	-

Column Name (RMS Table)	Default Value	Comments
RETAIL_ZONE_GROUP_ID	Lookup from price_zone_group table	Always sellable
COST_ZONE_GROUP_ID	NULL	Always non-orderable
STANDARD_UOM	EA	-
STORE_ORD_MULT	E	-
ORDERABLE_IND	N	-
INVENTORY_IND	Y	-
PACK_TYPE	NULL	Always non-orderable
ORDER_AS_TYPE	NULL	Always non-orderable
ITEM_AGGREGATE_IND	N	-
DIFF_1_AGGREGATE_IND	N	-
DIFF_2_AGGREGATE_IND	N	-
DIFF_3_AGGREGATE_IND	N	-
DIFF_4_AGGREGATE_IND	N	-
PRIMARY_REF_ITEM_IND	N	-
CONST_DIMEN_IND	N	-
GIFT_WRAP_IND	N	-
SHIP_ALONE_IND	N	-
ITEM_XFORM_IND	N	-
PACK_IND	Y	-
SIMPLE_PACK_IND	N	Always non-orderable
CATCH_WEIGHT_IND	N	Always non-orderable
ORDER_TYPE	NULL	Not catch weight
SALE_TYPE	NULL	Not catch weight
MERCHANDISE_IND	Y	-
FORECAST_IND	N	-
CONTAINS_INNER_IND	N	-
PERISHABLE_IND	N	-

Required file to load: dc_sellable_pack.dat

LOAD_PACK_COMPONENT

This function contains a PL/SQL block that selects from the DC_PACK_COMPONENT external 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 external Oracle 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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	
LAST_UPDATE_ID	Current user ID	
LAST_UPDATE_DATETIME	sysdate	

Required file to load: dc_pack_component.dat

LOAD_PACK_XREF

This function contains a PL/SQL block that selects from the DC_PACK_XREF and DC_PACK external tables and inserts the data to the RMS ITEM_MASTER table.

Most of the columns from the external Oracle 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_PACK_XREF and DC_PACK to ITEM_MASTER Column Defaults

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

Required file to load: dc_pack_xref.dat

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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_SELLABLE_PACK to PRICE_HIST Column Defaults

Column Name (RMS Table)	Default Value	Comments
ACTION_DATE	VDATE	-
TRAN_TYPE	0	-
LOC	0	-
REASON	0	-
SELLING_UNIT_RETAIL	Unit_retail	-
SELLING_UOM	EA	-

Required file to load: dc_sellable_pack.dat

INSERT_SELLABLE_RPM_IZP

This function selects from the DC_SELLABLE_PACK external 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. 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, 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	-
ITEM	Dc_sellable_pack.item	-
ZONE_ID	Rpm_zone.zone_id	For the department zone_group_id
STANDARD_RETAIL	Dc_sellable_pack.unit_retail	-
STANDARD_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
STANDARD_UOM	EA	-
SELLING_RETAIL	Dc_sellable_pack.unit_retail	-
SELLING_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
SELLING_UOM	EA	-
MULTI_UNIT_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id

Required file to load: dc_sellable_pack.dat

DEFAULT_PACKS

This function inserts item defaults from the merchandise hierarchy specifications for UDAs, VAT (if system_options.vat_ind = Y and 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 system_options.vat_ind = Y and 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 files to load: dc_orderable_pack.dat, dc_sellable_pack.dat

UPDATE_CATCH_WEIGHT_TYPE

This function updates the ITEM_MASTER table for those records that have been inserted by the LOAD_ORDERABLE_PACK function during the current run of DC_LOAD_PACKS.ksh.

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 files to load: dc_orderable_pack.dat

Item Supplier

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.

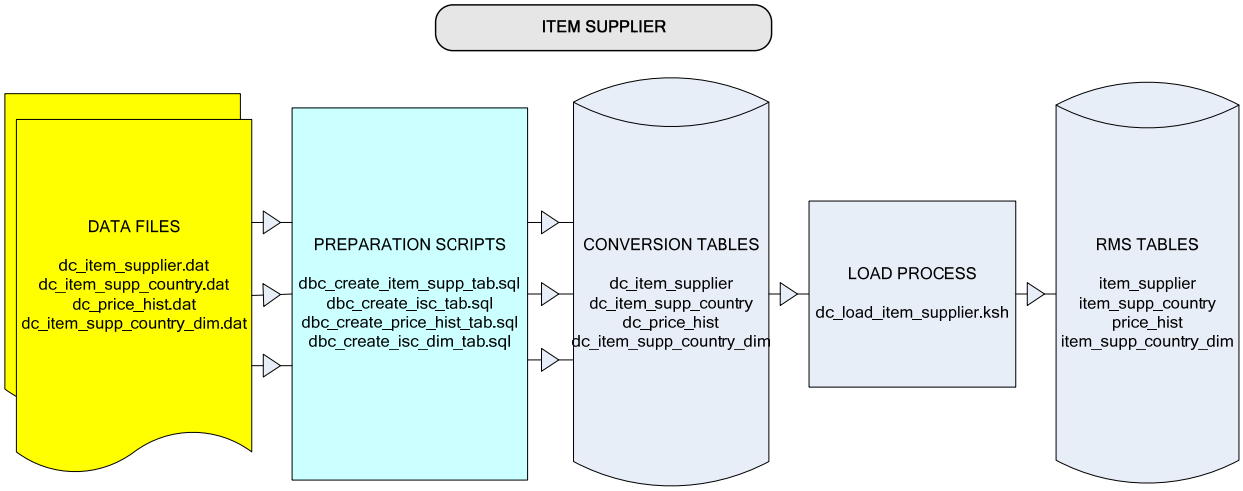
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_item_supplier.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_item_supplier_tab.sql
  - dbc_create_isc_tab.sql
  - dbc_create_ismc_tab.sql
  - dbc_create_isc_dim_tab.sql
  - dbc_create_price_hist_tab.sql
  - dbc_create_item_country_tab.sql
  - dbc_create_item_cost_head_tab.sql
  - dbc_create_item_cost_detail_tab.sql

Data Flow

The following diagram shows the 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 External Oracle 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_item_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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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**

Table create SQL script: **DBC_CREATE_ITEM_SUPPLIER_TAB.SQL**

External Oracle table created: **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 (sequence after dc_load_item_supplier.ksh:

- 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					EXTERNAL ORACLE 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					EXTERNAL ORACLE TABLE DEFINITION	
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)
CONCES SION_ 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)
CONSIGN MENT_ RATE	Numeric	12,4	N	Consignment rate for this item for the supplier. If the item is a consignment item, this field is required.	CONSIGN MENT_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**

Table create SQL script: **DBC_CREATE_ISC_TAB.SQL–** The SQL script is called by the external Oracle table created: **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 (sequence after dc_load_item_supplier.ksh:

- 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					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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					EXTERNAL ORACLE TABLE DEFINITION	
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					EXTERNAL ORACLE TABLE DEFINITION	
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					EXTERNAL ORACLE TABLE DEFINITION	
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)

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**

Table create SQL script: **DBC_ CREATE_ITEM_ISMC_TAB.SQL**

External Oracle table created: **DC_ITEM_SUPP_MANU_COUNTRY**

Suggested post-loading validation (sequence after dc_load_item_supplier.ksh:

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_ISC_DIM_TAB.SQL**

External Oracle table created: **DC_ITEM_SUPP_COUNTRY_DIM**

Suggested post-loading validation (sequence after dc_load_item_supplier.ksh:

- 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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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 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.	PRESEN TATION_ 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)

File Format					External Oracle Table Definition	
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)
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 NET_WEIGHT, WEIGHT_UOM, TARE_WEIGHT, and TARE_TYPE values.	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 WEIGHT, WEIGHT_ UOM, TARE_WEIGHT, and TARE_TYPE values.	NET_ WEIGHT	NUMBER(12,4)

File Format					External Oracle Table Definition	
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 WEIGHT, NET_WEIGHT, TARE_WEIGHT, and TARE_TYPE values. Default from System Options.	WEIGHT_UOM	VARCHAR2(4)
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 WEIGHT, NET_WEIGHT, WEIGHT_UOM, and TARE_TYPE values.	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 WEIGHT, NET_WEIGHT, WEIGHT_UOM, and TARE_WEIGHT values.	TARE_TYPE	VARCHAR2(6)

DC_ITEM_COUNTRY Table

File name: **DC_ITEM_COUNTRY.DAT**

Table create SQL script: **DBC_CREATE__COUNTRY_TAB.SQL**

The SQL script is called by the dc_load_item_supplier.ksh and will create the DC_ITEM_COUNTRY oracle external table

FILE FORMAT					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_ITEM_COST_HEAD_TAB.SQL**

The SQL script is called by the dc_load_item_supplier.ksh and will create the DC_ITEM_COST_HEAD oracle external table.

FILE FORMAT					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
NEGOTIAT ED_ITEM_C OST	Number	20,4	Y	This will hold the supplier negotiated item cost.	NEGOTIATED_ITE M_COST	NUMBER(20,4)
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**

Table create SQL script: **DBC_CREATE_ITEM_COST_DETAIL_TAB.SQL**

The SQL script is called by the dc_load_item_supplier.ksh and will create the DC_ITEM_COST_DETAIL oracle external table.

FILE FORMAT					EXTERNAL ORACLE 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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
DELIVERY_COUNTRY_ID	Alpha-numeric	3	Y	Country to which the item will be delivered to.	DELIVERY_COUNTRY_ID	VARCHAR2(3)
COND_TYPE	Alphanumeric	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

DC_LOAD_ITEM_SUPPLIER.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_item_supplier.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file.
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_ITEM_SUPPLIER

This function contains a PL/SQL block that selects from the DC_ITEM_SUPPLIER external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_ITEM_SUPPLIER to ITEM_SUPPLIER Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	SYSDATE	-
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	-

Required file to load: dc_item_supplier.dat

LOAD_ITEM_SUPP_COUNTRY

This function contains a PL/SQL block that selects from the DC_item_supp_country external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_ITEM_SUPP_COUNTRY to ITEM_SUPP_COUNTRY Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	-
LAST_UPDATE_DATETIME	sysdate	-
PRIMARY_SUPP_IND	from item_supplier	-
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.

Column Name (RMS Table)	Default Value	Comments
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
DEFAULT_UOP	Case	If NULL.
LEAD_TIME	from Sups	If NULL
COST_UOM	Standard uom from item_master	If NULL
CREATE_DATETIME	Sysdate	-
NEGOTIATED_ITEM_COST	The value will be taken from DC_ITEM_COST_HEAD.	-
EXTENDED_BASE_COST	The value will be taken from DC_ITEM_COST_HEAD.	-
INCLUSIVE_COST	The value will be taken from DC_ITEM_COST_HEAD.	-
BASE_COST	The value will be taken from DC_ITEM_COST_HEAD.	-

Required file to load: dc_item_supp_country.dat

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 external table and inserts the data to the RMS item_supp_manu_country table. All the columns from the external oracle table defined above will directly map to the RMS table. The function returns a Boolean value.

LOAD_ITEM_COUNTRY

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 external 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 external oracle table defined above will directly map to the RMS table.

LOAD_ITEM_COST_DETAIL

This function should do the following:

Insert the following columns into the item_cost_detail 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 external 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 external oracle table defined above will directly map to the RMS table.

INSERT_RPM_ITEM_ZONE_PRICE

This function selects from the DC_PRICE_HIST external 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	-
ITEM	Dc_price_hist.item	-
ZONE_ID	Rpm_zone.zone_id	For the department zone_group_id
STANDARD_RETAIL	Dc_price_hist.unit_retail	-
STANDARD_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
STANDARD_UOM	Dc_price_hist.uom	-
SELLING_RETAIL	Dc_price_hist.unit_retail	-
SELLING_ RETAIL_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id
SELLING_UOM	Dc_price_hist.uom	-
MULTI_UNIT_CURRENCY	Rpm_zone.currency_code	For the department zone_group_id

Required file to load: dc_price_hist.dat

LOAD_ITEM_SUPP_COUNTRY_DIM

This function contains a PL/SQL block that selects from the DC_ITEM_SUPP_COUNTRY_DIM external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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	-
LAST_UPDATE_DATETIME	SYSDATE	-
CREATE_DATETIME	SYSDATE	-
LWH_UOM	From SYSTEM_OPTIONS	If NULL
WEIGHT_UOM	From SYSTEM_OPTIONS	If NULL

Required file to load: dc_item_supp_country_dim.dat

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.

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:

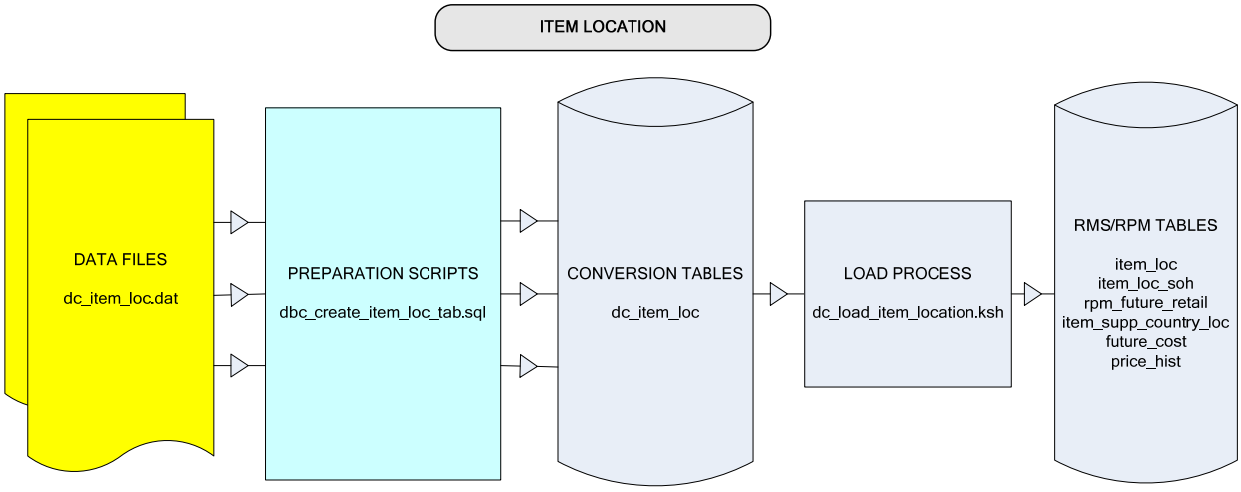
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. Refer to Chapter 2 for details.
- Segment load script dc_load_item_location.ksh

  This wrapper calls the external Oracle table create script dbc_create_item_loc_tab.sql.
- External Oracle table create script dbc_create_item_loc_tab.sql

Data Flow

The following diagram shows the 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 External Oracle 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_item_location.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle Table Definition columns Field Name, Data Type, and length define the physical external table.

DC_ITEM_LOC Table

File name: **DC_ITEM_LOC.DAT**

Table create SQL script: **DBC_CREATE_ITEM_LOC_TAB.SQL**

External Oracle table created: **DC_ITEM_LOC**

Suggested post-loading validation (sequence after dc_load_item_location.ksh:

- 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					EXTERNAL ORACLE 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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)
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)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
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)

DC_PRICE_HIST Table

File name: **DC_PRICE_HIST.DAT**

Table create SQL script: **DBC_CREATE_PRICE_HIST_TAB.SQL**

External Oracle table created: **DC_PRICE_HIST**

> **Note:** The DC_PRICE_HIST table must have rows/records for item levels that are transaction level or above. There cannot be any data for below-transaction-level items.

Suggested post-loading validation (sequence after dc_load_item_supplier.ksh:

- Capture counts from PRICE_HIST where PRICE_HIST.TRAN_TYPE = 0 and PRICE_HIST.REASON = 0 and PRICE_HIST.LOC = 0, and compare to flat file DC_PRIC_HIST.DAT to ensure that all rows are loaded.
- Ensure that PRICE_HIST.ITEM is a valid ITEM_MASTER.ITEM where ITEM_MASTER.ITEM_LEVEL <=ITEM.MASTER.TRAN_LEVEL.

FILE FORMAT					EXTERNAL ORACLE 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)
UNIT_ RETAIL	Numeric	20,4	Y	Item's current unit retail in the system's primary currency.	UNIT_RETAIL	NUMBER(20,4)
SELLING_ UOM	Alpha-numeric	4	Y	Item's current selling unit of measure for the item.	SELLING_UOM	VARCHAR2(4)

Required file to load: dc_item-cost_detail.dat

DC_LOAD_ITEM_LOCATION.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh and serves two purposes:

- It calls the create table scripts to create the external Oracle tables.
- It calls the load data script to insert data from the external Oracle tables to the RMS tables.

The script can be configured to create the tables and load data, or just load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_item_location.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_ITEM_LOC

This function contains a PL/SQL block that selects from the DC_ITEM_LOC external table and inserts the data to the RMS ITEM_LOC table. It joins the external 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.

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

The function returns a Boolean value.

DC_ITEM_LOC to ITEM_LOC Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	-
LAST_UPDATE_DATETIME	Sysdate	-
TAXABLE_IND	N	If NULL
CLEAR_IND	N	-

Column Name (RMS Table)	Default Value	Comments
STORE_PRICE_IND	N	-
RPM_IND	N	-
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	-
UNIT_RETAIL	selling_unit_retail	-
CREATE_DATETIME	Sysdate	-
STATUS_UPDATE_DATE	Sysdate	-
STATUS	A	-
LOCAL_ITEM_DESC	Default to ITEM_DESC	It will be populated with ITEM_MASTER.ITEM_DESC when it is null
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	
ITEM_GRANDPARENT	ITEM_MASTER.ITEM_GRANDPARENT	

Required file to load: dc_item_loc.dat

INSERT_ITEM_LOC_SOH

This function contains a PL/SQL block that selects from the DC_ITEM_LOC external table and inserts the data to the RMS ITEM_LOC_SOH table. It joins the external Oracle table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included. It joins the external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_ITEM_LOC to ITEM_LOC_SOH Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user id	-
LAST_UPDATE_DATETIME	Sysdate	-
CREATE_DATETIME	Sysdate	-
STOCK_ON_HAND	0	-
IN_TRANSIT_QTY	0	-
PACK_COMP_INTRAN	0	-
PACK_COMP_SOH	0	-
TSF_RESERVED_QTY	0	-
PACK_COMP_RESV	0	-
TSF_EXPECTED_QTY	0	-
PACK_COMP_EXP	0	-
RTV_QTY	0	-
NON_SELLABLE_QTY	0	-
CUSTOMER_RESV	0	-
CUSTOMER _BACKORDER	0	-
PACK_COMP_CUST_RESV	0	-
PACK_COMP_CUST_BACK	0	-
PACK_COMP_ NON_SELLABLE	0	-
ITEM_PARENT	ITEM_MASTER.ITEM_PARENT	-
ITEM_GRANDPARENT	ITEM_MASTER.ITEM_GRANDPARENT	-
AV_COST	ITEM_SUPP_COUNTRY.unit_cost of the primary supplier/primary country converted to location currency	-
PRIMARY_SUPP	ITEM_SUPP_COUNTRY.supplier with primary_supp_ind = Y for item	-
PRIMARY_CTRY	ITEM_SUPP_COUNTRY.origin_country_id with primary_supp_ind = Y and primary_country_ind = Y for item	-
AVERAGE_WEIGHT	NULL	Only defined if item is a simple pack catch weight item.

Required file to load: dc_item_loc.dat

INSERT_RPM_FUTURE_RETAIL

This function contains a PL/SQL block that selects from the DC_ITEM_LOC external table and inserts the data into the RPM RPM_FUTURE_RETAIL table.

Many of the columns from the external Oracle 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.

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

The function returns a Boolean value.

DC_ITEM_LOC to RPM_FUTURE_RETAIL Column Defaults

Column Name (RMS Table)	Default Value	Comments
FUTURE_RETAIL_ID	Sequence	-
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	-
ACTION_DATE	Vdate	-
ZONE_NODE_TYPE	If loc_type = 'S' then 0 If loc_type = 'W' then 2	-

Required file to load: dc_item_loc.dat

INSERT_RPM_FUTURE_RETAIL

This function should do the following:

Insert the following column values in RPM_STAGE_ITEM_LOC:

- stage_item_loc_id
- item
- loc
- loc_type
- selling_unit_retail
- selling_uom
- status
- create_date

This function selects from the DC_ ITEM_LOC external table joined with the item_master table and inserts the data to the RPM rpm_stage_item_loc table.

Warehouse locations are conditionally inserted into RPM_STAGE_ITEM_LOC based on the RPM system option, RECOGNIZE_WH_AS_LOCATIONS. Use 1 insert for stores and check this system option before the insert for warehouses. Warehouse must be stockholding locations.

Required file to load: dc_item_loc.dat

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 external 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 external Oracle table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included.

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

The function returns a Boolean value.

DC_ITEM_LOC to ITEM_SUPP_COUNTRY_LOC Column Defaults

Column Name (RMS Table)	Default Value	Comments
LAST_UPDATE_ID	Current user ID	-
LAST_UPDATE_DATETIME	SYSDATE	-
PICKUP_LEAD_TIME	LEAD_TIME	If NULL
CREATE_DATETIME	SYSDATE	-

Column Name (RMS Table)	Default Value	Comments
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	-
ROUND_TO_INNER_PCT	ITEM_SUPP_COUNTRY. ROUND_ TO_INNER_PCT	-
ROUND_TO_CASE_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_CASE_PCT	-
ROUND_TO_LAYER_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_LAYER_PCT	-
ROUND_TO_PALLET_PCT	ITEM_SUPP_COUNTRY. ROUND_TO_PALLET_PCT	-

Required file to load: dc_item_loc.dat

INSERT_FUTURE_COST

This function selects from the DC_ITEM_LOC external 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 external table with a virtual table that is a union of store and warehouse, so that only stockholding warehouses are included.

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

The function returns a Boolean value.

DC_ITEM_LOC to FUTURE_COST Column Defaults

Column Name (RMS Table)	Default Value	Comments
ACTIVE_DATE	VDATE	-
START_IND	Y	-
CALC_DATE	VDATE	-

Required file to load: dc_item_loc.dat

INSERT_PRICE_HIST

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 external 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 external 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.

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

The function returns a Boolean value.

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	-
TRAN_TYPE	0	-
REASON	0	-

Required file to load: dc_item_loc.dat

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 system_options.vat_ind is Y and default_tax_type is not GTAX)
- ITEM_SEASONS
- ITEM_TICKET

The following programs are included in this functional area:

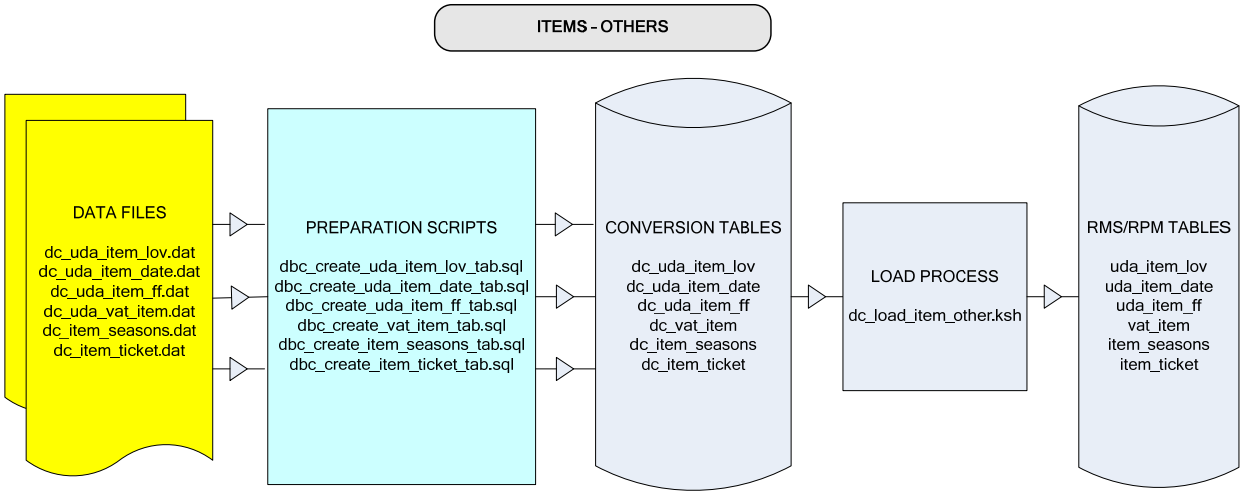
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.
- Segment load script dc_load_item_other.ksh

  This wrapper calls the external Oracle table create and load scripts listed below.
- External Oracle table create scripts:
  - dbc_create_uda_item_lov.sql
  - dbc_create_uda_item_date.sql
  - dbc_create_uda_item_ff.sql
  - dbc_create_vat_item.sql
  - dbc_create_item_seasons.sql
  - dbc_create_item_ticket.sql

Data Flow

The following diagram shows the data flow for the Items–Others 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 External Oracle 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 created in the order listed.

The dc_load_item_other.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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**

Table create SQL script: **DBC_CREATE_UDA_ITEM_LOV_TAB.SQL**

External Oracle table created: **DC_UDA_ITEM_LOV**

Suggested post-loading validation (sequence after dc_load_item_other.ksh:

- 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					EXTERNAL ORACLE 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)
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**

Table create SQL script: **DBC_CREATE_UDA_ITEM_DATE_TAB.SQL**

External Oracle table created: **DC_UDA_ITEM_DATE**

Suggested post-loading validation (sequence after dc_load_item_other.ksh:

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_UDA_ITEM_FF_TAB.SQL**

External Oracle table created: **DC_UDA_ITEM_FF**

Suggested post-loading validation (sequence after dc_load_item_other.ksh:

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_VAT_ITEM_TAB.SQL**

External Oracle table created: **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					EXTERNAL ORACLE 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

DC_ITEM_SEASONS Table

File name: **DC_ITEM_SEASONS.DA**T

Table create SQL script: **DBC_CREATE_ITEM_SEASONS_TAB.SQL**

External Oracle table created: **DC_ITEM_SEASONS**

Suggested post-loading validation (sequence after dc_load_item_other.ksh:

- 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					EXTERNAL ORACLE 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**

Table create SQL script: **DBC_CREATE_ITEM_TICKET_TAB.SQL**

External Oracle table created: **DC_ITEM_TICKET**

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

- 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					EXTERNAL ORACLE 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_LOAD_ITEM_OTHER.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh. It serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just to load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_item_other.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topics describe the load functions that are included in the load script.

LOAD_UDA_ITEM_LOV

This function contains a PL/SQL block that selects from the DC_UDA_ITEM_LOV external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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

LOAD_UDA_ITEM_DATE

This function contains a PL/SQL block that selects from the DC_UDA_ITEM_DATE external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

DC_UDA_ITEM_DATE to UDA_ITEM_DATE 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_date.dat

LOAD_UDA_ITEM_FF

This function contains a PL/SQL block that selects from the DC_UDA_ITEM_FF external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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

LOAD_VAT_ITEM

This function contains a PL/SQL block that selects from the DC_VAT_ITEM external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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.

Column Name (RMS Table)	Default Value	Comments
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

LOAD_ITEM_SEASONS

This function contains a PL/SQL block that selects from the DC_ITEM_SEASONS external 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 (external table field values are NULL or not defined).

The function returns a Boolean value.

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

LOAD_ITEM_TICKET

This function contains a PL/SQL block that selects from the DC_ITEM_TICKET external table and inserts the data to the RMS ITEM_TICKET table.

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

The function returns a Boolean value.

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

8

Multiple Sets of Books (MSOB)

This chapter describes the 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:

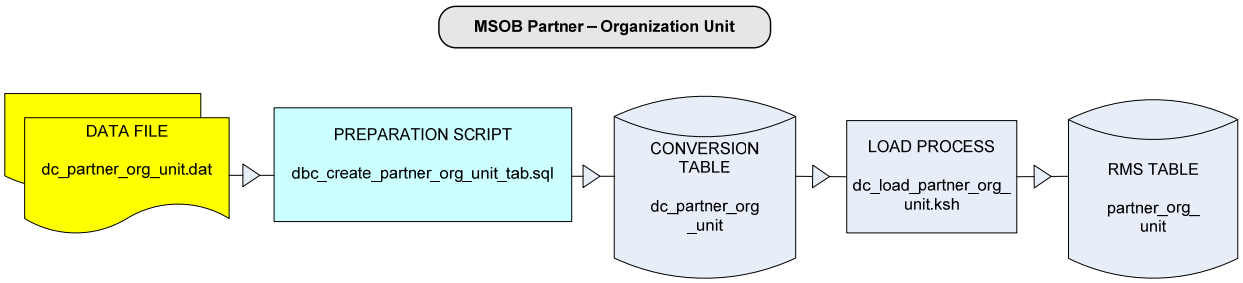
- Main wrapper script dc_load_main.ksh

  This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.
- Segment load script dc_load_partner_org_unit.ksh

  This wrapper calls the external Oracle table create and load script listed below.
- External Oracle table create scripts: dbc_create_partner_org_unit_tab.sql

Data Flow

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



File Format and External Oracle 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_wh_org.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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**

Table create SQL script: **DBC_CREATE_PARTNER_ORG_UNIT_TAB.SQL**

External Oracle table created: **DC_PARTNER_ORG_UNIT**

File Format					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

DC_LOAD_PARTNER_ORG_UNIT.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh. It serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just to load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

LOAD_PARTNER_ORG_UNIT

This function contains a PL/SQL block that selects from the DC_PARTNER_ORG_UNIT external table and inserts the data to the RMS PARTNER_ORG_UNIT table. All the columns from the external Oracle table defined previously map directly to the RMS table.

The following fields are required:

- PARTNER
- ORG_UNIT_ID
- PARTNER_TYPE

The function returns a Boolean value.

Required file to load: dc_partner_org_unit.dat

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:

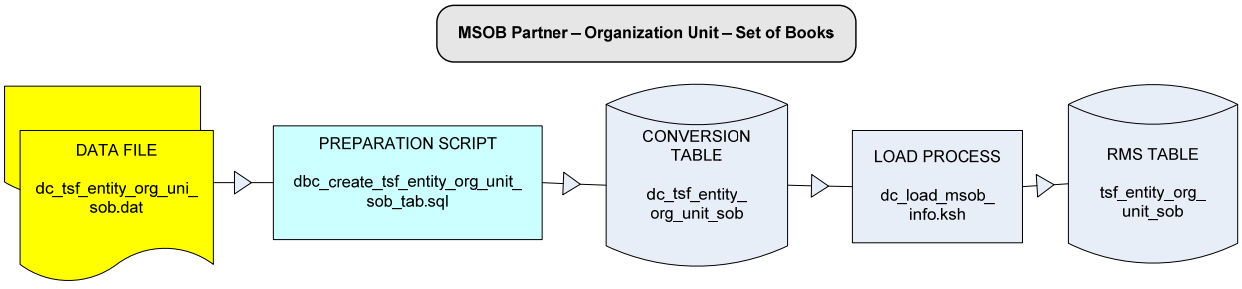
- Main wrapper script dc_load_msob_info.ksh

  This main script is used across all functional areas to call segment load scripts. See Chapter 2, "Master Script (DC_LOAD_MAIN.KSH)," for details.
- Segment load script dc_load_msob.info.ksh

  This wrapper calls the external Oracle table create and load script listed below.
- External Oracle table create scripts: dbc_create_tsf_entity_org_unit_sob_tab.sql

Data Flow

The following diagram shows the data flow for the MSOB Transfer Entity – Organization Unit – Set of Books functional area:



File Format and External Oracle 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_wh_org.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.

External Oracle Table Definition

In the table definitions that follow, the External Oracle 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**

Table create SQL script: **DBC_CREATE_TSF_ENTITY_ORG_UNIT_SOB_TAB.SQL**

External Oracle table created: **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					External Oracle Table Definition	
Field Name	**Data Type**	**Max Length**	**Req'd**	**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)

DC_LOAD_LOAD_MSOB_INFO.KSH Segment Wrapper / Load Script

This ksh script is called by the master script dc_load_main.ksh. It serves two purposes:

- It calls the create table scripts to create external Oracle tables.
- It calls the load data script to insert data from external Oracle tables to RMS tables.

The script can be configured to create the tables and load data, or just to load data at run time. The create table scripts are called only if a parameter option (-c) is passed from the command line. By default (without the option -c), this script loads the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

LOAD_TSF_ENTITY_ORG_UNIT_SOB

This function contains a PL/SQL block that selects from the DC_TSF_ENTITY_ORG_UNIT external table and inserts the data to the RMS TSF_ENTITY_ORG_UNIT_SOB table. All the columns from the external Oracle table defined previously map directly to the RMS table.

All fields are required:

The function returns a Boolean value.

Required file to load: dc_partner_org_unit.dat

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
- STOP_SHIP
- TRANSIT_TIMES

Merchandise Hierarchy Tables

There is no additional data to be loaded manually.

Organizational Hierarchy Tables

- STORE_ATTRIBUTES
- WH_ATTRIBUTES
- STORE_SHIP_DATE
- WH_DEPT
- WH_DEPT_EXPL
- WH_BLACKOUT
- WH_STORE_ASSIGN

Supplier Tables

- SUP_ATTRIBUTES
- SUP_INV_MGMT
- SUP_REPL_DAY
- SUPP_PREISSUE
- SUPS_MIN_FAIL

Items Tables

- PACK_TMPL_HEAD
- PACK_TMPL_DETAIL
- ITEM_ATTRIBUTES
- ITEM_SUPP_UOM
- ITEM_LOC_TRAITS
- SUB_ITEMS_HEAD
- SUB_ITEMS_DETAIL
- ITEM_FORECAST
- REPL_ITEM_LOC
- REPL_DAY
- MASTER_REPL_ATTR

10

Brazil Localization

This chapter describes data conversion for the Brazil localization extension of the following base entities, listed in the order that they must be loaded:

- WH
- STORE
- SUPS
- ITEM_COUNTRY
- ENTITY_CNAE_CODES
- PARTNER

Brazil Localization data conversion scripts load Brazil-specific fiscal attributes for the business entities to RMS. Brazil fiscal attributes are defined as L10N flexible attributes in RMS utilizing RMS's LFAS infra-structure (Localization Flexible Attribute Solution). Prior to running the Brazil-specific data conversion scripts, flex attributes must be defined for the business entity (for example, Item-Country) for the country of Brazil; staging tables (for example, STG_ITEM_COUNTRY_L10N_EXT_BR) must be generated based on the flex attribute definition.

The Brazil-specific data conversion scripts will first load the data in the .dat file into the corresponding staging tables and then call a base RMS package function (L10N_FLEX_API_SQL.INSERT_L10N_EXT_TBL) to move the data from the staging tables to the corresponding base RMS entity extension tables (for example, ITEM_COUNTRY_L10N_EXT) based on the flex field definition.

DC_ENTITY_CNHAE_CODES_BR.DAT Table

File name: **DC_ITEM_SUPP_COUNTRY.DAT**

Table create SQL script: **DBC_CREATE_ENTITY_CNAE_CODES_BR_TAB.SQL –** The SQL script is called by the external Oracle table created: **DC_ENTITY_CNAE_CODES_BR**

FILE FORMAT					**EXTERNAL ORACLE TABLE DEFINITION**	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
KEY_ VALUE_1	Alpha-numeric	10	Y	Contains the unique supplier/store/warehouse/company/outside location/partner.	KEY_VALUE_1	VARCHAR2(10)
KEY_ VALUE_2	Alpha-numeric	6	Y	This will contain the type of module when the module is PTNR/OUTLOC else it will be null.	KEY_VALUE_2	VARCHAR2(6)
MODULE	Alpha-numeric	4	Y	Contains the type of the module. SUP for Supplier, PTNR for partner.	MODULE	VARCHAR2(4)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
COUNTRY_ID	Alpha-numeric	3	Y	Contains the unique code that identifies the country where the entity belongs.	COUNTRY_ID	VARCHAR2(3)
CNAE_CODE	Alpha-numeric	9	Y	These are the set of codes published by the fiscal authorities. Certain taxes are calculated by MTR on basis of these codes.	CNAE_CODE	VARCHAR2(9)
PRIMARY_IND	Alphanumeric	1	Y	Y means it is a primary cnae code and N means it is not a primary cnae code.	PRIMARY_IND	VARCHAR2(1)

Required file to load: dc_entity_cnae_codes_br.dat

DC_LOAD_ENTITY_CNAE_CODES_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables.
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

LOAD_ENTITY_CNAE_CODES_BR

This function inserts the following column values in ENTITY_CNAE_CODES:

- key_value_1
- key_value_2
- module
- country_id
- cnae_code
- primary_ind

This function selects from the DC_ENTITY_CNAE_CODES_BR and loads directly into the RMS entity_cnae_codes table. All the columns from the external oracle table defined above will directly map to the RMS table. The function returns a Boolean value.

DC_ITEM_COUNTRY_L10N_EXT_BR.DAT Table

File name: **DC_ITEM_COUNTRY_L10N_EXT_BR.DAT**

Table create SQL script: **DBC_CREATE_ITEM_COUNTRY_L10N_EXT_BR_TAB.SQL –** The SQL script is called by the external Oracle table created: **DC_ITEM_COUNTRY_L10N_EXT_BR**

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
ITEM	Alpha-numeric	25	Y	Alphanumeric value which identifies the item.	ITEM	VARCHAR2(25)
COUNTRY_ID	Alpha-numeric	3	Y	Id of the Country	COUNTRY_ID	VARCHAR2(3)
SERVICE_IND	Alpha-numeric	1	N	Service Item	SERVICE_IND	VARCHAR2(1)
ORIGIN_CODE	Alpha-numeric	6	N	Merchandise Origin	ORIGIN_CODE	VARCHAR2(6)
CLASSIFICATION_ID	Alpha-numeric	25	N	NCM	CLASSIFICATION_ID	VARCHAR2(25)
NCM_CHAR_CODE	Alpha-numeric	25	N	NCM Characteristic	NCM_CHAR_CODE	VARCHAR2(25)
EX_IPI	Alpha-numeric	25	N	EX IPI	EX_IPI	VARCHAR2(25)
PAUTA_CODE	Alphanumeric	25	N	Pauta	PAUTA_CODE	VARCHAR2(25)
SERVICE_CODE	Alpha-numeric	25	N	Service Code	SERVICE_CODE	VARCHAR2(25)
FEDERAL_SERVICE	Alpha-numeric	25	N	Federal Service	FEDERAL_SERVICE	VARCHAR2(25)

Required file to load: dc_item_country_l10n_ext_br.dat

DC_LOAD_ITEM_COUNTRY_L10N_EXT_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

LOAD_ITEM_COUNTRY_L10N_EXT_BR

This function should insert the following column values in Brazil-specific staging table STG_ITEM_COUNTRY_L10N_EXT_BR and move the data from the staging table to the base RMS L10N extension table (ITEM_COUNTRY_L10N_EXT):

- process_id
- item
- country_id
- service_ind
- origin_code
- classification_id
- ncm_char_code
- ex_ipi
- pauta_code
- service_code
- federal_service

This function selects from the DC_ITEM_COUNTRY_L10N_EXT_BR and loads directly into the Brazil-specific staging table stg_item_country_l10n_ext_br table. It then calls function L10N_FLEX_API_SQL.INSERT_L10N_EXT_TBL to move the data from the staging table to the base RMS L10N extension table (ITEM_COUNTRY_L10N_EXT). All the columns from the external oracle table defined above will directly map to the RMS table. The table below lists columns that do not exist on DC_ITEM_COUNTRY_L10N_EXT_BR and the defaults to be used for them. The function returns a Boolean value.

Column Name (RMS Table)	Default Value	Comments
PROCESS_ID		A sequence C_ITEM_COUNTRY_L10N_EXT_SEQ defined in the script will insert the values into process_id column.

DC_PARTNER_L10N_EXT_BR.DAT Table

File name: **DC_PARTNER_L10N_EXT_BR.DAT**

Table create SQL script: **DBC_CREATE_PARTNER_L10N_EXT_BR_TAB.SQL –** The SQL script is called by the external Oracle table created: **DC_PARTNER_L10N_EXT_BR.**

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
PARTNER_TYPE	Alpha-numeric	6	Y	Specifies the type of partner.	PARTNER_TYPE	VARCHAR2(6)
PARTNER_ID	Alpha-numeric	10	Y	Unique identifying number for a partner within the system	PARTNER_ID	VARCHAR2(10)
TAXPAYER_TYPE	Alpha-numeric	1	N	Taxpayer type	TAXPAYER_TYPE	VARCHAR2(1)
ADDR_1	Alpha-numeric	240	N	Address	ADDR_1	VARCHAR2(240)
ADDR_2	Alpha-numeric	240	N	Address	ADDR_2	VARCHAR2(240)
ADDR_3	Alpha-numeric	240	N	Address	ADDR_3	VARCHAR2(240)
NEIGHBORHOOD	Alpha-numeric	240	N	Neighborhood	NEIGHBORHOOD	VARCHAR2(240)
JURISDICTION_CODE	Alpha-numeric	10	N	City	JURISDICTION_CODE	VARCHAR2(10)
STATE	Alpha-numeric	5	N	State	STATE	VARCHAR2(5)
COUNTRY	Alpha-numeric	3	N	Country	COUNTRY	VARCHAR2(3)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
POSTAL_ CODE	Alpha-numeric	30	N	Postal Code	POSTAL_CODE	VARCHAR2(30)
CPF	Alpha-numeric	11	N	Individual Taxpayer	CPF	VARCHAR2(11)
CNPJ	Alpha-numeric	14	N	Corporate Taxpayer	CNPJ	VARCHAR2(14)
NIT	Alpha-numeric	250	N	NIT	NIT	VARCHAR2(250)
SUFRAMA	Alpha-numeric	250	N	SUFRAMA	SUFRAMA	VARCHAR2(250)
IM	Alpha-numeric	250	N	City Inscription	IM	VARCHAR2(250)
IE	Alpha-numeric	250	N	State Inscription	IE	VARCHAR2(250)
IPI_IND	Alpha-numeric	1	N	IPI Contributor	IPI_IND	VARCHAR2(1)
ICMS_ CONTRIB_ IND	Alpha-numeric	1	N	ICMS Contributor	ICMS_ CONTRIB_IND	VARCHAR2(1)

Required file to load: dc_ partner_l10n_ext_br.dat

DC_LOAD_PARTNER_L10N_EXT_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

This function selects from the DC_PARTNER_L10N_EXT_BR and loads directly into the RMS stg_partner_l10n_ext_br table. All the columns from the external oracle table defined above will directly map to the RMS table. The table below lists columns that do not exist on DC_PARTNER_L10N_EXT_BR and the defaults to be used for them The function returns a Boolean value.

Column Name (RMS Table)	Default Value	Comments
PROCESS_ID		A sequence C_PARTNER_L10N_EXT_SEQ defined in the script will take care of inserting the values into process_id column.
PIS_CONTRIB_IND		Yes or No
COFINS_CONTRIB_IND		Yes or No

DC_STORE_ADD_L10N_EXT_BR.DAT Table

File name: **DC_STORE_ADD_L10N_EXT_BR.DAT**

Table create SQL script: **DBC_CREATE_STORE_ADD_L10N_EXT_BR_TAB.SQL –** The SQL script is called by the external Oracle table created: **DC_STORE_ADD_L10N_EXT_BR**

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
STORE	Number	10	Y	Store	STORE	NUMBER(10)
TAXPAYER_TYPE	Alpha-numeric	1	N	Taxpayer type	TAXPAYER_TYPE	VARCHAR2(1)
ADDR_1	Alpha-numeric	240	N	Address	ADDR_1	VARCHAR2(240)
ADDR_2	Alpha-numeric	240	N	Address	ADDR_2	VARCHAR2(240)
ADDR_3	Alpha-numeric	240	N	Address	ADDR_3	VARCHAR2(240)
NEIGHBORHOOD	Alpha-numeric	240	N	Neighborhood	NEIGHBORHOOD	VARCHAR2(240)
JURISDICTION_CODE	Alpha-numeric	10	N	City	JURISDICTION_CODE	VARCHAR2(10)
STATE	Alpha-numeric	5	N	State	STATE	VARCHAR2(5)
COUNTRY	Alpha-numeric	3	N	Country	COUNTRY	VARCHAR2(3)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
POSTAL_CODE	Alpha-numeric	30	N	Postal Code	POSTAL_CODE	VARCHAR2(30)
CPF	Alpha-numeric	11	N	Individual Taxpayer	CPF	VARCHAR2(11)
CNPJ	Alpha-numeric	14	N	Corporate Taxpayer	CNPJ	VARCHAR2(14)
NIT	Alpha-numeric	250	N	NIT	NIT	VARCHAR2(250)
SUFRAMA	Alpha-numeric	250	N	SUFRAMA	SUFRAMA	VARCHAR2(250)
IM	Alpha-numeric	250	N	City Inscription	IM	VARCHAR2(250)
IE	Alpha-numeric	250	N	State Inscription	IE	VARCHAR2(250)
ISS_CONTRIB_IND	Alpha-numeric	1	N	ISS Contributor	ISS_CONTRIB_IND	VARCHAR2(1)
RURAL_PROD_IND	Alpha-numeric	1	N	Rural Producer	RURAL_PROD_IND	VARCHAR2(1)
IPI_IND	Alphanumeric	1	N	IPI Contributor	IPI_IND	VARCHAR2(1)
ICMS_CONTRIB_IND	Alpha-numeric	1	N	ICMS Contributor	ICMS_CONTRIB_IND	VARCHAR2(1)
MATCH_OPR_TYPE	Alpha-numeric	6	N	Matching Operation Type	MATCH_OPR_TYPE	VARCHAR2(6)
CONTROL_REC_ST_IND	Alpha-numeric	1	N	Control Recovery of ST	CONTROL_REC_ST_IND	VARCHAR2(1)

Required file to load: dc_store_add_l10n_ext_br.dat

DC_LOAD_STORE_ADD_L10N_EXT_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

This function selects from the DC_STORE_ADD_L10N_EXT_BR and loads directly into the RMS stg_store_add_l10n_ext_br table. All the columns from the external oracle table defined above will directly map to the RMS table. The table below lists columns that do not exist on DC_STORE_ADD_L10N_EXT_BR and the defaults to be used for them. The function returns a Boolean value.

Column Name (RMS Table)	Default Value	Comments
PROCESS_ID		A sequence C_STORE_ADD_L10N_EXT_SEQ defined in the script will take care of inserting the values into the process_id column.
PIS_CONTRIB_IND		Yes or No
COFINS_CONTRIB_IND		Yes or No

DC_SUPS_L10N_EXT_BR.DAT Table

File name: **DC_SUPS_L10N_EXT_BR.DAT**

Table create SQL script: **DBC_ CREATE_SUPS_L10N_EXT_BR_TAB.SQL –** The SQL script is called by the external Oracle table created: **DC_SUPS_L10N_EXT_BR**

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
SUPPLIER	Number	10	Y	Supplier	SUPPLIER	NUMBER(10)
TAXPAYER_TYPE	Alpha-numeric	1	N	Taxpayer type	TAXPAYER_TYPE	VARCHAR2(1)
ADDR_1	Alpha-numeric	240	N	Address	ADDR_1	VARCHAR2(240)
ADDR_2	Alpha-numeric	240	N	Address	ADDR_2	VARCHAR2(240)
ADDR_3	Alpha-numeric	240	N	Address	ADDR_3	VARCHAR2(240)
NEIGHBORHOOD	Alpha-numeric	240	N	Neighborhood	NEIGHBORHOOD	VARCHAR2(240)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
JURISDICTION_CODE	Alpha-numeric	10	N	City	JURISDICTION_CODE	VARCHAR2(10)
STATE	Alpha-numeric	5	N	State	STATE	VARCHAR2(5)
COUNTRY	Alpha-numeric	3	N	Country	COUNTRY	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	Postal Code	POSTAL_CODE	VARCHAR2(30)
CPF	Alpha-numeric	11	N	Individual Taxpayer	CPF	VARCHAR2(11)
CNPJ	Alpha-numeric	14	N	Corporate Taxpayer	CNPJ	VARCHAR2(14)
NIT	Alpha-numeric	250	N	NIT	NIT	VARCHAR2(250)
SUFRAMA	Alpha-numeric	250	N	SUFRAMA	SUFRAMA	VARCHAR2(250)
IM	Alphanumeric	250	N	City Inscription	IM	VARCHAR2(250)
IE	Alphanumeric	250	N	State Inscription	IE	VARCHAR2(250)
ISS_CONTRIB_IND	Alpha-numeric	1	N	ISS Contributor	ISS_CONTRIB_IND	VARCHAR2(1)
SIMPLES_IND	Alpha-numeric	1	N	SIMPLES Contributor	SIMPLES_IND	VARCHAR2(1)
ST_CONTRIB_IND	Alpha-numeric	1	N	ST Contributor	ST_CONTRIB_IND	VARCHAR2(1)
RURAL_PROD_IND	Alpha-numeric	1	N	Rural Producer	RURAL_PROD_IND	VARCHAR2(1)
IPI_IND	Alpha-numeric	1	N	IPI Contributor	IPI_IND	VARCHAR2(1)
ICMS_CONTRIB_IND	Alpha-numeric	1	N	ICMS Contributor	ICMS_CONTRIB_IND	VARCHAR2(1)
PIS_CONTRIB_IND	Alpha-numeric	1	N	PIS Contributor	PIS_CONTRIB_IND	VARCHAR2(1)
COFINS_CONTRIB_IND	Alpha-numeric	1	N	Cofins Contributor	COFINS_CONTRIB_IND	VARCHAR2(1)

Required file to load: dc_sups_l10n_ext_br.dat

DC_LOAD_SUPS_L10N_EXT_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

This function selects from the DC_SUPS_L10N_EXT_BR and loads directly into the RMS stg_sups_l10n_ext_br table. All the columns from the external oracle table defined above will directly map to the RMS table. . The table below lists columns that do not exist on DC_SUPS_L10N_EXT_BR and the defaults to be used for them. The function returns a Boolean value.

Column Name (RMS Table)	Default Value	Comments
PROCESS_ID		A sequence C_SUPS_L10N_EXT_SEQ defined in the script will insert the values into process_id column.

DC_WH_L10N_EXT_BR.DAT Table

File name: **DC_WH_L10N_EXT_BR.DAT**

Table create SQL script: **DBC_CREATE_WH_L10N_EXT_BR_TAB –** The SQL script is called by the external Oracle table created: **DC_WH_L10N_EXT_BR**

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
Field Name	**Data Type**	**Max Length**	**Req. Ind**	**Description**	**Field Name**	**Data Type**
WH	Number	10	Y	Warehouse	WH	NUMBER(10)
TAXPAYER_TYPE	Alpha-numeric	1	N	Taxpayer type	TAXPAYER_TYPE	VARCHAR2(1)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
ADDR_1	Alpha-numeric	240	N	Address	ADDR_1	VARCHAR2(240)
ADDR_2	Alpha-numeric	240	N	Address	ADDR_2	VARCHAR2(240)
ADDR_3	Alpha-numeric	240	N	Address	ADDR_3	VARCHAR2(240)
NEIGHBORHOOD	Alpha-numeric	240	N	Neighborhood	NEIGHBORHOOD	VARCHAR2(240)
JURISDICTION_CODE	Alpha-numeric	10	N	City	JURISDICTION_CODE	VARCHAR2(10)
STATE	Alpha-numeric	5	N	State	STATE	VARCHAR2(5)
COUNTRY	Alpha-numeric	3	N	Country	COUNTRY	VARCHAR2(3)
POSTAL_CODE	Alpha-numeric	30	N	Postal Code	POSTAL_CODE	VARCHAR2(30)
CPF	Alpha-numeric	11	N	Individual Taxpayer	CPF	VARCHAR2(11)
CNPJ	Alpha-numeric	14	N	Corporate Taxpayer	CNPJ	VARCHAR2(14)
NIT	Alpha-numeric	250	N	NIT	NIT	VARCHAR2(250)
SUFRAMA	Alpha-numeric	250	N	SUFRAMA	SUFRAMA	VARCHAR2(250)
IM	Alpha-numeric	250	N	City Inscription	IM	VARCHAR2(250)
IE	Alphanumeric	250	N	State Inscription	IE	VARCHAR2(250)
ISS_CONTRIB_IND	Alpha-numeric	1	N	ISS Contributor	ISS_CONTRIB_IND	VARCHAR2(1)
RURAL_PROD_IND	Alpha-numeric	1	N	Rural Producer	RURAL_PROD_IND	VARCHAR2(1)
IPI_IND	Alpha-numeric	3	N	IPI Contributor	IPI_IND	VARCHAR2(3)
ICMS_CONTRIB_IND	Alpha-numeric	1	N	ICMS Contributor	ICMS_CONTRIB_IND	VARCHAR2(1)
MATCH_OPR_TYPE	Alphanumeric	6	N	Matching Operation Type	MATCH_OPR_TYPE	VARCHAR2(6)

FILE FORMAT					EXTERNAL ORACLE TABLE DEFINITION	
CONTROL_REC_ST_IND	Alpha-numeric	1	N	Control Recovery of ST	CONTROL_REC_ST_IND	VARCHAR2(1)

Required file to load: dc_wh_l10n_ext_br.dat

DC_LOAD_WH_L10N_EXT_BR

This ksh script will be called by the dc_load_main.ksh and serves two purposes:

- Call the create table scripts to create the Oracle external tables
- Call the load data script to insert data from the Oracle external tables to the RMS tables.

The script calls internal functions (defined within the script) that insert-select from the external tables to the RMS tables. The script can be configured to create the tables and load data or just load data at run time. The create table scripts will only be called if a parameter option (-c) is passed in from the command line. By default (without the option -c) this script will load the data.

The dc_load_wh_org.ksh script utilizes a common library file and configuration file. The library file contains functions common across all segment wrapper scripts. The configuration file defines the directories used by the data conversion scripts.

For a specific function to process the load, the script checks a status file. If the data file is valid, one unique status file (*.status) is generated per function, if it does not yet exist, to signal that a load has started. If the file already exists, the script skips the load and writes a message to the log file. For the data file to be valid, it must satisfy these requirements:

- It must exist in the data directory defined in the common configuration file
- It must allow read access.
- It must contain information (have a size greater than 0).

The next topic describes the load functions that are included in the load script.

This function selects from the DC_WH_L10N_EXT_BR and loads directly into the RMS stg_wh_l10n_ext_br table. All the columns from the external oracle table defined above will directly map to the RMS table. The table below lists columns that do not exist on DC_WH_L10N_EXT_BR and the defaults to be used for them. The function returns a Boolean value.

Column Name (RMS Table)	Default Value	Comments
PROCESS_ID		A sequence, C_WH_L10N_EXT_SEQ, defined in the script will take care of inserting the values into process_id column.
PIS_CONTRIB_IND		Yes or No
COFINS_CONTRIB_IND		Yes or No

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 that 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 these tables:	RIB_ERRORS RIB_LANG RIB_TYPE_SETTINGS RIB_SETTINGS
	Calls RIB_DOCTYPES.SQL, which launches a SQL loader session to insert into these tables:	RIB_DOCTYPES
RMSUOM.SQL		UOM_CLASS
RMSCOUNTRIES.SQL		COUNTRY
RMSSTATES.SQL		STATE
RMSCURRENCIES.SQL		CURRENCIES
STATICIN.SQL	Inserts directly into these tables:	SYSTEM_OPTIONS ADD_TYPE ADD_TYPE_MODULE COST_CHG_REASON DUMMY DEAL_COMP_TYPE DOC_LINK DYNAMIC_HIER_CODE INV_STATUS_TYPES INV_STATUS_CODES LANG MC_REJECTION_REASONS ORDER_TYPES PRICE_ZONE_GROUP 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 these tables:	CVB_HEAD ELC_COMP
	Calls GENERAL_DATA_INSTALL_SQL VAT_CODE_REGION to insert into these tables:	VAT_REGION VAT_CODES VAT_CODE_RATES
	Calls GENERAL_DATA_INSTALL_SQL ADD_TYPE to insert into this table:	ADD_TYPE
	Calls GENERAL_DATA_INSTALL_SQ ADD_TYPE_MODULE to insert into this table:	ADD_TYPE_MODULE
	Calls GENERAL_DATA_INSTALL. UNIT_OPTIONS to insert into this table:	UNIT_OPTIONS
	Calls GENERAL_DATA_INSTALL_SQL ELC_COMP_EXPENSES to insert into this table:	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 this table:	ELC_COMP
ADD_FILTER_POLICY .SQL	Calls the DBMS_RLS.ADD_POLICY function to implement fine-grained access control.	
NAVIGATE.SQL		NAV_ELEMENT NAV_SERVER NAV_COMPONENT EVIEW NAV_ICON
NAVROLE.SQL		NAV_ELEMENT_MODE_ROLE

Script Name	Scripts/Packages Called	Tables Inserted
CODES.SQL		CODE_HEAD CODE_DETAIL CODE_DETAIL_TRANS
POPULATE_SEC_ FORM_ ACTION.SQL		SEC_FORM_ACTION
RESTART.SQL		RESTART_PROGRAM_STATUS RESTART_CONTROL+C11
RTK_ERRORS.SQL		RTK_ERRORS
RTK_REPORTS.SQL		RTK_REPORTS
TL_COLUMNS.SQL		TL_COLUMNS
WIZARD.SQL		WIZARD_TEXT
CONTEXT.SQL		CONTEXT_HELP
POPULATE_FORM_ LINKS.SQL		FORM_LINKS
POPULATE_FORM_ LINKS_ROLE.SQL		FORM_LINKS_ROLE
UOM_X_ CONVERSION.SQL		UOM_X_CONVERSION
VAR_UPC_EAN_ LOAD.SQL		VAR_UPC_EAN
MULTIVIEW_DATA. SQL		MULTIVIEW_SAVED_45 MULTIVIEW_DEFAULT_45
RMSUOMCONV1.SQL		UOM_CONVERSION
RMSUOMCONV2.SQL		UOM_CONVERSION
CALENDAR.SQL		HALF CALENDAR SYSTEM_VARIABLES PERIOD
SA_SYSTEM_ REQUIRED.SQL	Calls SA_METADATA.SQL to insert into these 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 this table:	SA_REALM_TYPE
	Calls SA_METADATA.SQL, which calls SA_REALM.SQL to insert into this table:	SA_REALM

Script Name	Scripts/Packages Called	Tables Inserted
	Calls SA_METADATA.SQL, which calls SA_PARM_TYPE.SQL to insert into this table:	SA_PARM_TYPE
	Calls SA_METADATA.SQL, which calls SA_PARM.SQL to insert into this table:	SA_PARM
RMS12RTM.SQL	Calls ENTRY_TYPE.SQL to insert into this table:	ENTRY_TYPE
	Calls ENTRY_STATUS.SQL to insert into this table:	ENTRY_STATUS
	Calls OGA.SQL to insert into this table:	OGA
	Calls TARIFF_TREATMENT.SQL to insert into this table:	TARIFF_TREATMENT
	QUOTA_CATEGORY.SQL	QUOTA_CATEGORY
	Calls COUNTRY_TARIFF_ TREATMENT.SQL to insert into this table:	COUNTRY_TARIFF_ TREATMENT
	Calls HTS_HEADINGS.SQL to insert into this table:	HTS_CHAPTER
BASE_FORM_MENU_ ELEMENTS.SQL	Calls all the XML.SQL scripts for each form present in the application to insert into these tables:	FORM_ELEMENTS FORM_ELEMENTS_TEMP FORM_ELEMENTS_LANGS_ TEMP FORM_MENU_LINK MENU_ELEMENTS MENU_ELEMENTS_TEMP MENU_ELEMENTS_LANGS_ TEMP