

Retek® Data Warehouse 10.2



Operations Guide



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

No part of this documentation may be reproduced or transmitted in any form or by any means without the express written permission of Retek Inc., Retek on the Mall, 950 Nicollet Mall, Minneapolis, MN 55403.

Information in this documentation is subject to change without notice.

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

Corporate Headquarters:

Retek Inc.
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403
888.61.RETEK (toll free US)
+1 612 587 5000

European Headquarters:

Retek
110 Wigmore Street
London
W1U 3RW
United Kingdom
Switchboard:
+44 (0)20 7563 4600
Sales Enquiries:
+44 (0)20 7563 46 46
Fax: +44 (0)20 7563 46 10

Retek[®] Data Warehouse[™] is a trademark of Retek Inc.

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

This unpublished work is protected by confidentiality agreement, and by trade secret, copyright, and other laws.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

©2002 Retek Inc. All rights reserved.

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

Printed in the United States of America.



Customer Support

Customer Support hours:

Customer Support is available 7x24x365 via e-mail, phone, and Web access.

Depending on the Support option chosen by a particular client (Standard, Plus, or Premium), the times that certain services are delivered may be restricted. Severity 1 (Critical) issues are addressed on a 7x24 basis and receive continuous attention until resolved, for all clients on active maintenance.

Contact Method Contact Information

Internet (ROCS) www.retek.com/support
Retek's secure client Web site to update and view issues

E-mail support@rettek.com

Phone US & Canada: 1-800-61-RETEK (1-800-617-3835)
World: +1 612-587-5800
EMEA: 011 44 1223 703 444
Asia Pacific: 61 425 792 927

Mail Retek Customer Support
Retek on the Mall
950 Nicollet Mall
Minneapolis, MN 55403

When contacting Customer Support, please provide:

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

Contents

Chapter 1 – Introduction.....	1
What is RDW and data warehousing?.....	2
Technical architecture	3
Dimension processing.....	4
Fact processing	5
Process to update records in RDW	6
Where you can find more information	6
Chapter 2 – Dimension data concepts	7
An overview of RDW dimension processing.....	7
Dimensions in RDW 10.2.....	9
Major changes and lower-level dimensions	10
Minor changes and top-level dimensions	11
Actions during processing.....	11
Maintenance columns in the DM table.....	12
Keys and identifiers.....	12
Next_key_val.....	12
As-was vs. as-is	13
Pushdowns	13
An overview of RDW dimension processing flows.....	13
Data preparation for lower-level dimensions	14
Data preparation for lower-level dimensions flow description	16
Major and minor change capture for lower-level dimensions flow	17
Major and minor change capture for lower-level dimensions flow description	19
Processing for the top-level dimensions.....	20
Top-level processing data flow description.....	22
Datamart table	22
Dimension datamart (DM) table.....	23

Chapter 3 – Fact data concepts	25
An overview of RDW fact processing	25
Fact functional areas.....	27
Fact table types: base and aggregate	28
General fact processing	29
Detailed fact load description.....	30
Base fact loading process flow description	32
Fact aggregation	33
Positional fact aggregation	33
Standard fact aggregation.....	35
Fact process flow—fact aggregation description	36
Derived datamarts.....	38
Chapter 4 – Compression and partitioning.....	41
Overview of compression.....	41
What compression does	41
The mechanics of compression.....	42
Compressed tables and CUR tables.....	43
Coping with major changes	44
Partitioning for the Oracle client only	45
Overview of partitioning strategies	45
Implementing RDW partitioning.....	46
Partitioning strategy and requirements for MicroStrategy 7	48
An example of setup and maintenance for partitioning and warehouse partition mapping for compressed inventory tables	50
How Oracle implements partitions	53
Summary.....	54
Chapter 5 – RDW program overview	55
Program features	55
Program return code	55
Program status control files	55
Restart and recovery	56
Message logging.....	58
Program error file	59
DWI reject files	60
Schema files.....	61
Resource files	61
Command line parameters	62
Partitioning	63

Typical run and debugging situations	63
DWI dimension extract.....	64
DWI fact extract	65
RDW dimension load	68
RDW fact load.....	69

Chapter 6 – RDW interfaces 71

Retek Merchandising System.....	73
Dimension data.....	73
Fact data.....	73
Retek Sales Audit.....	74
Retek TopPlan.....	75
Retek Customer Order Management.....	75
Client-supplied data.....	76

Chapter 7 – Program flow diagrams 77

Batch scheduling	77
Setting up the batch schedule	77
dwi_config.env settings.....	78
rdw_config.env settings	78
RMS, ReSA and the RDW batch schedule	79
TopPlan to RDW scheduling.....	79
Data from undefined sources.....	79
RDW batch schedule for DB2 clients only	80
Program flow diagrams	80
Legend: RDW 10.2 dimension programs.....	81
Legend: RDW 10.2 fact programs.....	88

Chapter 8 – Program reference lists..... 97

Dimension programs	97
Fact programs.....	112
Maintenance programs	130
Program type and operation type descriptions	136
Dimension types	136
Fact types.....	142
Maintenance types.....	149

**Appendix A – Application programming interface (API) flat
file specifications 151**

API format.....	151
File layout.....	151
General business rules and standards common to all APIs	152
Dimensions.....	153
Extraction and load.....	153
Load only.....	203
Facts	221
Extraction and load.....	221
Load only.....	280

Chapter 1 – Introduction

Retek Data Warehouse (RDW) version 10.2 contains enhanced localization functionality and works in conjunction with the Retek Extract Transform and Load (RETL) 10.2 framework. This architecture optimizes a high performance data processing tool that lets database batch processes take advantage of parallel processing capabilities. In addition, RDW can be extended beyond its traditional reliance upon Oracle to IBM's DB2 Universal Database (UDB) and NCR's Teradata.

With the implementation of RETL, the RDW client benefits from the following capabilities:

- Database Independence: Allows RDW to be deployed on different database platforms
- Parallel computing technology:
 - Promotes the flexibility of a stand-alone solution
 - Lets database batch processes take full advantage of parallel processing capabilities
 - Increases scalability, leveraging parallel processing of both the system and database server (reads, writes, performs transformations and aggregations)
- Expanded use of Application Programming Interfaces (API): Allows for easier customization
- Elimination of table triggers: Reduces the burden on the source system
- Extensible Markup Language (XML) scripts: Facilitate the framework's ability to process fact and dimension data by using valid operators
- Streamlined ETL Code: Provides for less data storage, easier implementation, and reduced maintenance requirements through decreased code volume and complexity

What is RDW and data warehousing?

A data warehouse is a physical place, a database, where you can place data from a transactional system, such as Retek Merchandising System (RMS), for the purpose of querying that data. In order to work with RDW, you start by populating it with existing data from source systems such as RMS, Retek Sales Audit (ReSA), and Retek TopPlan.

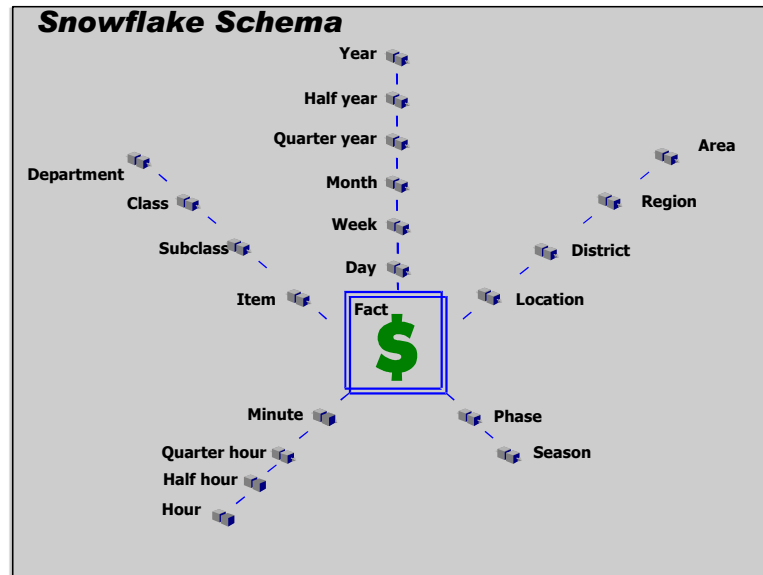
RDW uses very sophisticated techniques to populate the data warehouse. Explained in greater detail throughout this guide, these techniques include the use of batch programs (usually called ‘modules’ here) that extract data from source systems like RMS and then rapidly transform and load it into the warehouse. Techniques used to load data into the warehouse vary depending upon whether the data consists of ‘facts’ or ‘dimensions’.

Understanding the differences between fact and dimension data depends first upon understanding data processing in a data warehouse. RDW uses an online analytical processing (OLAP) application that serves as an interface to your data, giving it meaning through pre-designed and custom queries and reports. The data warehouse itself supports these queries by structuring data in a useful schema. Note that the word ‘schema’ in this context is an industry-standard term that refers to the way in which data is modeled and organized throughout a data warehouse and should not be confused with the ‘schema files’ that are described later in this document. (More information about schema files can be found in the RETL 10.2 Programmer’s Guide.)

At the center of this schema is fact data. Facts are the transactions that occur in your data warehouse’s source systems, such as RMS. You might want to look at sales transaction facts, or inventory stock count facts at stores or warehouses, or inventory movement facts.

Facts have little meaning by themselves because they are usually just values, for example: 6 sales at a store, 15 items left at a warehouse, or 300 items transferred. What gives fact data true meaning in RDW is the intersection of dimensions in which facts exist. In other words, 6 sales on Wednesday at store B, or 15 dishwashers in stock last Monday at the Chicago warehouse, or 300 blouses transferred during the last week in February from the St. Louis warehouse to the Denver warehouse. Dimension data, therefore, exists in the data warehouse to serve as reference data to facts.

The schema of a data warehouse illustrates its data elements and their inter-relationships. The following graphic describes the schema used in RDW:



Snowflake schema in RDW 10.2

RDW's schema, the 'snowflake schema', starts out as a star with a fact in the middle surrounded by rays pointing out from the center. These points are the dimension data that give meaning to the fact by serving as points of reference.

RDW contains far greater volumes of fact data than it does dimension data. Besides being more abundant than dimensions, facts change constantly as new data enters the database. Dimension data, on the other hand, changes much less frequently. New stores need to be added into the data warehouse much less frequently than new sales transactions (fact data) that need to be processed daily. Because of the different natures of fact and dimension data, RDW employs different techniques to load and manipulate the data.

The dimension and fact processing sections located later in this chapter illustrate the differences in these two processes, both of which contribute to the success of RDW as your data warehouse. A more detailed description of dimension and fact processing concepts continues throughout the next two chapters.

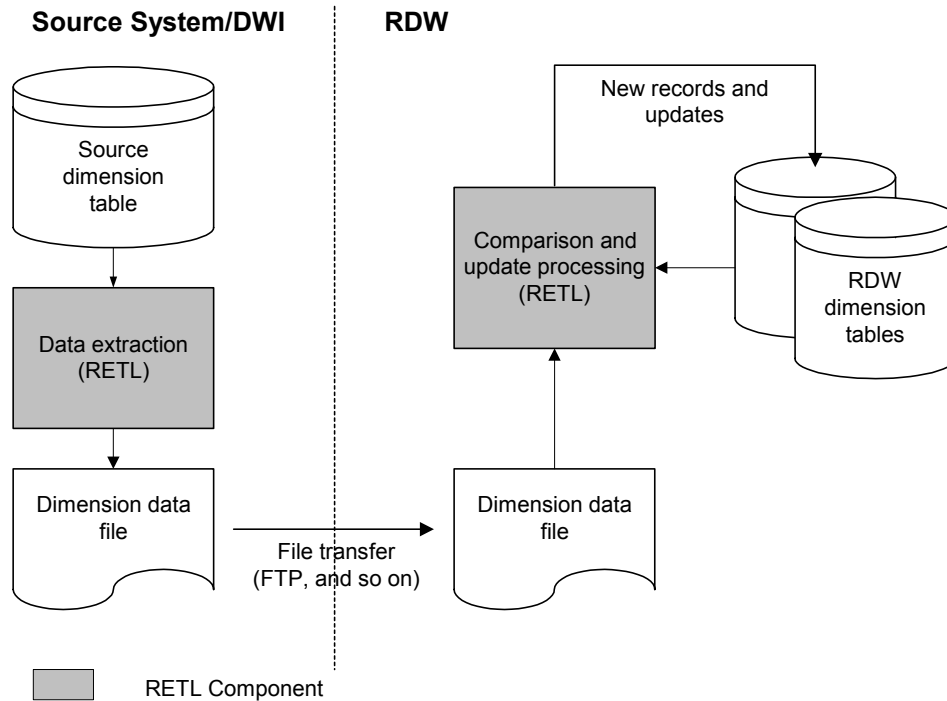
Technical architecture

The primary goal of the RETL architecture is to take advantage of enhanced parallel processing capabilities and at the same time provide a database independent solution that runs streamlined code. The RETL framework runs and parses through the valid operators composed in XML scripts.

In this section, three features of RDW 10.2 are described: dimension processing, fact processing, and the process to update records.

Dimension processing

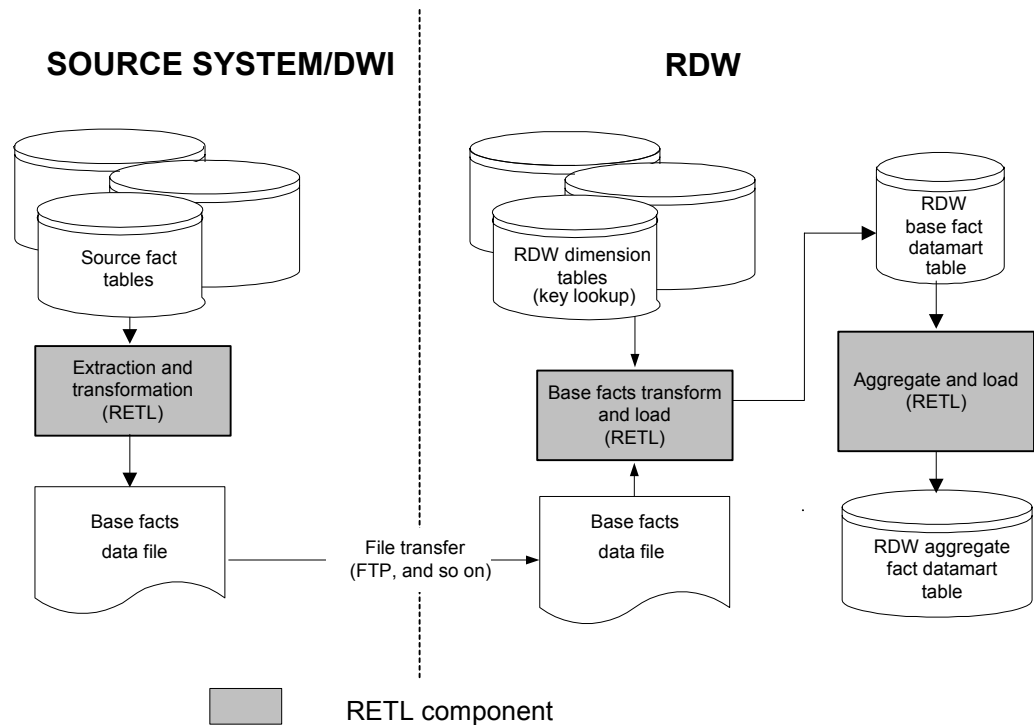
The following diagram illustrates the dimension processing architecture that is employed in RDW 10.2. The process involves extracting the current dimension data (as a snapshot of the entire applicable table) from the source system and comparing it with the historical data in RDW. This comparison eliminates the need to capture frequent dimension changes as they occur in the source system over the course of the day. The comparison is performed on the RETL framework and written back directly to the datamart tables in RDW.



Dimension processing in RDW 10.2

Fact processing

The following diagram illustrates fact processing in RDW 10.2. Because of improvements in DWI functionality, the RETL component is responsible for extracting the data from the source system. Only the changed and/or new facts for today are extracted. The data is then transformed, aggregated, and loaded directly into RDW without the need for staging tables.



Fact processing in RDW 10.2

Process to update records in RDW

Because RETL does not currently support a database update operator, the actual updates into the database are accomplished through one of two processes, depending upon whether a normal update or an incremental update is occurring. A normal update is one that uses incoming records to replace old records in the target table. An incremental update (applicable to fact processing only) is one that sums the incoming records with the old records in the target table and replaces those old records with the new summed records.

Note: The temporary tables that are mentioned throughout this operations guide are always dropped by the batch code every day, after the various batch processes that use the temporary tables complete.

Normal update description

- 1 The dataset (containing the new records) is written into a temporary table.
- 2 This temporary table is used to determine which of the old update records in the target table should be deleted.
- 3 The old records are deleted from the target table.
- 4 The new records are inserted into the target table.

Incremental update description (applicable to fact processing only)

- 1 The dataset (containing the new records) is written to a temporary table.
- 2 The records to be updated are read from the target table and a second temporary table (temporary table 2) is created.
- 3 The temporary table 2 is used to determine which of the old update records in the target table should be deleted, and those records are deleted.
- 4 The records in the temporary table and in temporary table 2 are combined to form a new dataset.
- 5 The new dataset is grouped by the primary keys of the target table to sum up the required fact fields.
- 6 The resulting dataset is written to the target table (that is, the records are inserted into the target table).

Where you can find more information

You can find more information about RDW 10.2 in these resources:

- RDW 10.2 Data Model
- RDW 10.2 Installation Guides
- RETL 10.2 Programmer's Guide
- RDW 10.2 User Guide
- RDW online help

Chapter 2 – Dimension data concepts

This chapter describes how RDW processes dimension data from the source system or systems. This chapter presents the following dimension data concepts in RDW 10.2:

- An overview of dimension data processing
- The dimensions in RDW 10.2
- Detailed dimension processing flows

An overview of RDW dimension processing

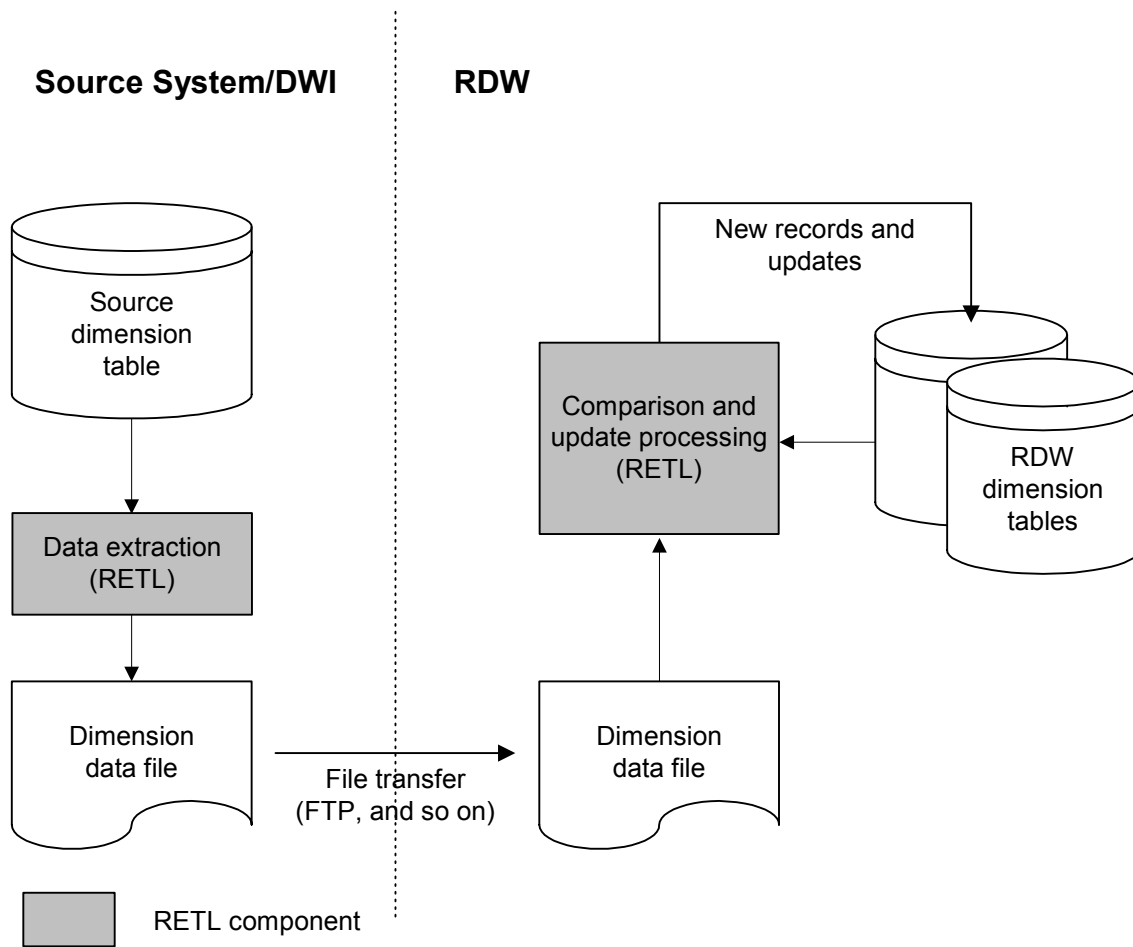
The following description and “Dimension processing in RDW 10.2” diagram offer an overview of RDW’s dimension process.

The process involves the extraction of the source system data as a snapshot of the applicable table.

The data file is transferred to the RDW server using a common data transfer process such as FTP.

The data undergoes a comparison and update processing. This comparison eliminates the need to capture frequent dimension changes as they occur in the source system over the course of the day. The comparison is performed on the RETL framework and written back directly to the datamart tables in RDW.

Note the role that RETL plays, extracting the data and performing comparison and update processing.



Dimension processing in RDW 10.2

Dimensions in RDW 10.2

RDW consists of the dimensions shown in this table. Product and organization are bold because they can be subject to (what is called in RDW) a ‘major change’. A further discussion of this concept follows.

RDW 10.2 Dimensions	
Organization and Product dimensions can be reclassified (major-changed).	
Company	Competitor
Currency Code	Customer Account
Customer and Customer Demographic	Customer and Product Clustering
Customer Geographic	Employee
Item-Location Trait Cross-dimension	Item-Supplier-Location Cross-dimension
Market Data	Organization
Plan Season	Product
Product Season	Promotion
Reason	Regionality
Register	ReSA Total Type
Retail Type	Sub-transaction Type
Supplier	Tender Type
Time (time calendar, time of day, time like for like)	Voucher Age Band

Major changes and lower-level dimensions

A major change occurs whenever an entity changes its place in the product hierarchy (group, department, and item can be reclassified) or in the organization hierarchy (area, region, district, and location can be reclassified). This type of reclassification alters the relationship among entities in a hierarchy. Of the dimensions, only product and organization can undergo a major change, and they are known as lower-level dimensions. Another way to think of these is as ‘dimensions with major changeable lower levels’. Because product and organization are aggregating dimensions, a major change results in an altered data aggregation within their hierarchy.

The history of an entity before and after the major change can be tracked and compared. For example, suppose an item is moved from one subclass to another within its product hierarchy of department and class. While there are many good reasons for a retailer to move, or reclassify, an item in this way—perhaps there is a need to track that item in relation to different items in the system—RDW still needs to track sales for that item from its new location in the product hierarchy...both before and after the change. (See the sections, “Pushdowns” and “As-was vs. as-is”, later in this chapter.) Looking at the diagram, “Dimension processing in RDW 10.2” (located at the beginning of this chapter), you can see the box labeled “Compare and update processing”. Major change processing occurs at this point. RDW handles major changes by assigning the reclassified item, to use the same example, a new surrogate key. The surrogate key, along with the dimension’s identifier, lets RDW track the dimension and all transactions related to it at any point in time.

Minor changes and top-level dimensions

A minor change means that an attribute of an entity is changed, but its position in the hierarchy remains the same.

The dimensions that can *only* undergo minor changes are known as top-level dimensions and consist of every dimension except organization and product. The levels of the top-level dimensions cannot be reclassified; they are static. Note that product and organization dimensions *can undergo* minor changes, but minor changes are not significant enough to alter their hierarchies.

One example of a minor change is the modification of a description field in a dimension. For example, a description of a subclass is changed from “Humorous Cards” to “Funny Cards”. This type of change does not alter the relationship of subclass to any other level of the hierarchy above or below it. The record is simply updated to reflect the description change; a new surrogate key does not need to be inserted. Minor change dimension processing in RDW is less complex than major change processing.

Actions during processing

During the actual processing of data, there are four kinds of actions that can happen to a dimensional entity in the RDW:

- **Insert:** When an entity is created, it is inserted into the system.
- **Major Change:** When a major change occurs, an entity is effectively closed and re-inserted, so that its history before and after the change can be tracked and compared. (See the passages, “Pushdowns” and “As-was vs. as-is”, later in this chapter.)
- **Minor Change:** When an entity undergoes a minor change, the attribute of the entity is changed, but its position in the hierarchy remains the same.
- **Close:** When an entity is no longer active, it is considered to be closed. Although closing an entity in a transactional system often involves deleting it from the system entirely, in an analytical system like RDW, the entity’s record is retained so that its history can continue to be reported. One exception in RDW is dimensional matrices, where only the current relationship between two source system identifiers (and their surrogate keys) is kept (for example, item_key and itemlst_key on the PROD_ITEMLST_MTX_DM table). Note, however, that in the case of one specific dimension matrix table, PACK_ITEM_MATRIX, the history of closed reclasses is kept.

Maintenance columns in the DM table

- **dm_recd_last_updt_dt**: The last date on which this record was either inserted, updated, or closed.
- **dm_recd_load_dt**: The date on which this record was loaded/created.
- **dm_recd_close_dt**: The last date on which this record could be considered active. Closes occur either because of a record being deleted in the source system, or because a record had a major change applied to it. If the record is an active dimensional record, it will have a default value of '4444-04-04' as a dm_recd_close_dt.
- **dm_recd_curr_flag**: Indicates whether a record can be considered active. Valid values are 'Y'es and 'N'o.

Keys and identifiers

Most dimensional entities in the RDW have both keys (typically referred to as 'surrogate keys' or 'pseudokeys') and identifiers (typically abbreviated 'idnt'). The term 'identifier' in the RDW refers to the identifier given to the entity when it was created in the source system. However, in the RDW, this identifier cannot always be used to uniquely identify an entity. An entity may undergo a major change, where it is closed and reloaded in order to mark the change in hierarchy, so that history can be tracked before and after the change. It may also be deleted in the source system, and its identifier reused later. Both of these situations result in multiple records in the RDW tables for the same entity. In order to distinguish between different states of the same entity, or different entities with the same identifier, the RDW must use some other value to uniquely mark it. A surrogate key is a unique value used to identify an entity in the RDW. A new key is attached to an entity whenever it is inserted into a datamart dimension table.

Next_key_val

Each datamart dimension table which needs a surrogate key has a record on the table MAINT_DIM_KEY_DM. This record holds the next valid surrogate key for the dimension. The dimension's load program queries this record at the beginning of its run, and, at the end of its run, updates the record with the next valid key for the next run. Note that there are some cases in which the identifiers in the source system are unique, and they will not change over time. If there is no need in RDW to keep track of the changes, RDW does not always create surrogate keys in the applicable dimension tables (ORG_LOC_TRAIT_DM, for example).

As-was vs. as-is

One of the primary types of analysis in the RDW is drilling, that is, seeing a particular report at a given level, and then being able to see the same report at a lower level to examine data at a finer level of granularity. This type of analysis makes well-defined hierarchies extremely important in the RDW. Drill paths must be clear, and facts must add up between levels of aggregation. This requirement explains why changes in an entity's place in the hierarchy are considered major.

One of the effects of a major change is that the presence of two surrogate keys makes it possible to compare an entity's performance before and after it undergoes a major change. Fact aggregate tables are also left in a state where the data ties out, because all history was summed up under the entity's old key, while all future data will be summed up under its new key. This is referred to as *as-was reporting*, because history is seen as part of the hierarchy it was in. In order to achieve *as-is reporting*, in which history is shown as if it had occurred under the new hierarchy, fact aggregate tables would either have to be eliminated (resulting in poor report performance) or would have to be rebuilt to account for the hierarchical changes. *RDW only supports as-was reporting.*

Pushdowns

In order to optimize performance, each datamart dimension table holds the keys and identifiers of its parent in the hierarchy, its parent's parent, and so on. Because of this structure, when an entity at a higher level undergoes a major change, all of its descendents (held within the lower levels of the hierarchy) must undergo the major change with it. The same rule applies for closes. Each lower-level dimension program joins with that dimension's immediate parent table to get parent keys for incoming data to compare with the keys in the dimension table to decide if there is a major change. For instance, if a group changes to another division, the group key is changed. The incoming department data joins with the group dimension table to get the group key for that department and group combination. If the department's group key is different than the group key in the department dimension table, a major change is recognized. The pushdown effect is seen after each lower-level dimension program runs individually.

An overview of RDW dimension processing flows

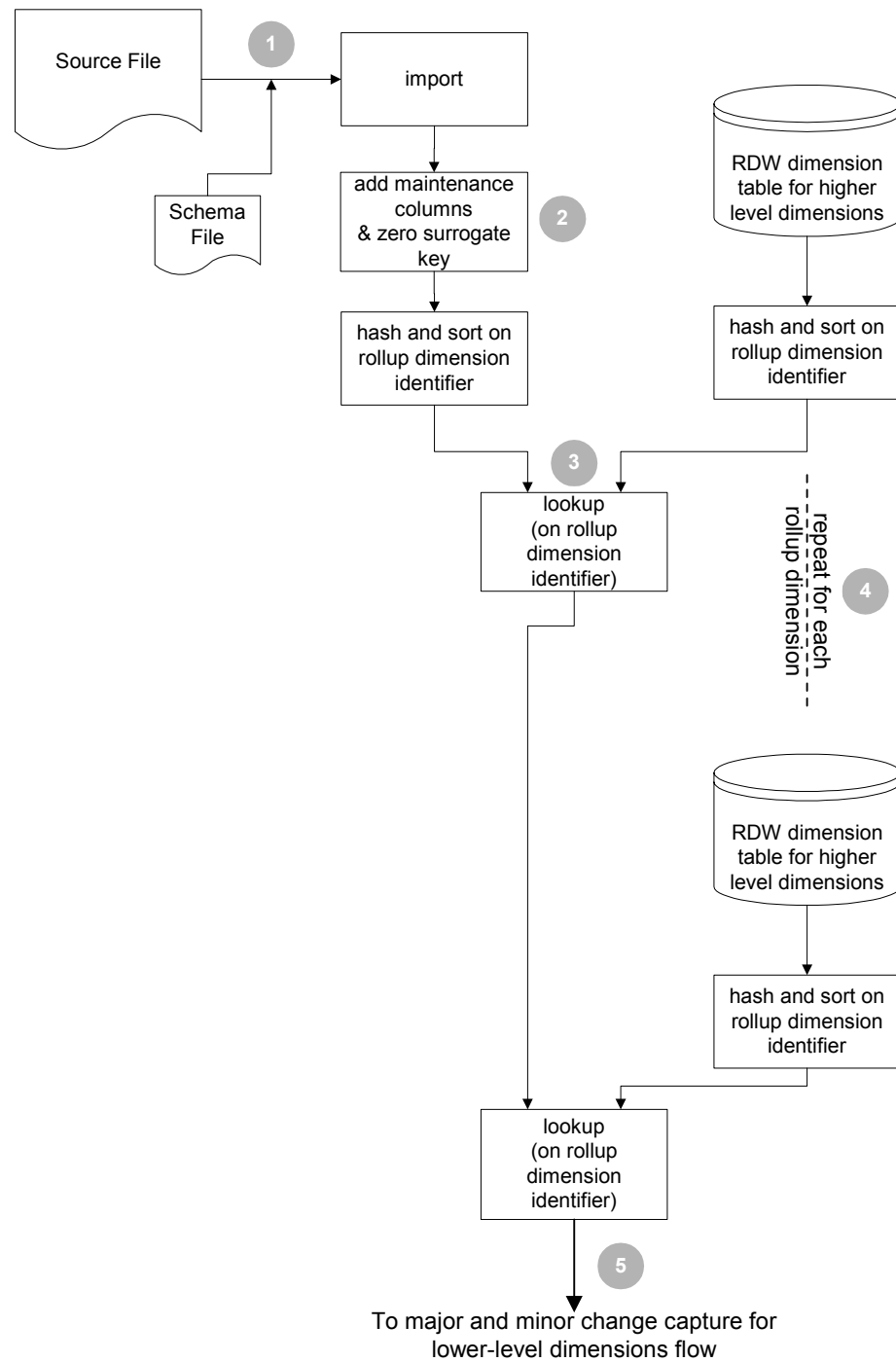
The remainder of this chapter illustrates the flow of dimension data from source tables to RDW datamart tables. The processing described begins with a dimension text file provided from the source system. That file is read by dimension and maintenance RDW libraries, which load the data to the dimensional datamart table. Each dimension processing module will have a record (or entry) on the maintenance table PROGRAM_CONTROL_DM, with values populated in the operation_type and program_type columns. See Chapter 8, "Program reference lists", for more details.

Data preparation for lower-level dimensions

The flow diagram in this section—“Data preparation for lower-level dimensions flow”—is used to produce the source data that is used in step one (1) of the flow that follows, “Major and minor change capture for lower-level dimensions flow.” In other words, this flow is not a separate process, but is the predecessor of the flow that immediately follows. Together, the two flows represent a single process that updates a lower-level dimension.

This flow only applies to dimensions that have a parent dimension table above them. That is, they are not the highest-level dimensions in the hierarchy. The source data stream in the change compare dataset must match the RDW table structure, and it must have all the higher-level keys to be able to detect major changes and to have the necessary fields to produce insert records. Because the dimension data in the source system, RMS for example, is typically normalized, it contains only the idnt of the immediate rollup dimensions, not of any higher-level dimensions. To get all the idnts and keys of all higher-level dimensions (denormalized for performance in RDW), the incoming data is joined with all the immediate rollup dimension tables from RDW. To ensure that the most recent information is being used (and thus to account for major changes in higher level dimensions), the order in which the dimension updating process is applied is to start with the highest-level dimensions and to work down the hierarchy until the base level dimension is processed. Thus, the higher-level RDW tables that are used in the joins will have already been refreshed with the incoming data for those dimensions.

The diagram below shows this flow. Explanations of each numbered item on the diagram follow it.



Data preparation for lower-level dimensions flow

Data preparation for lower-level dimensions flow description

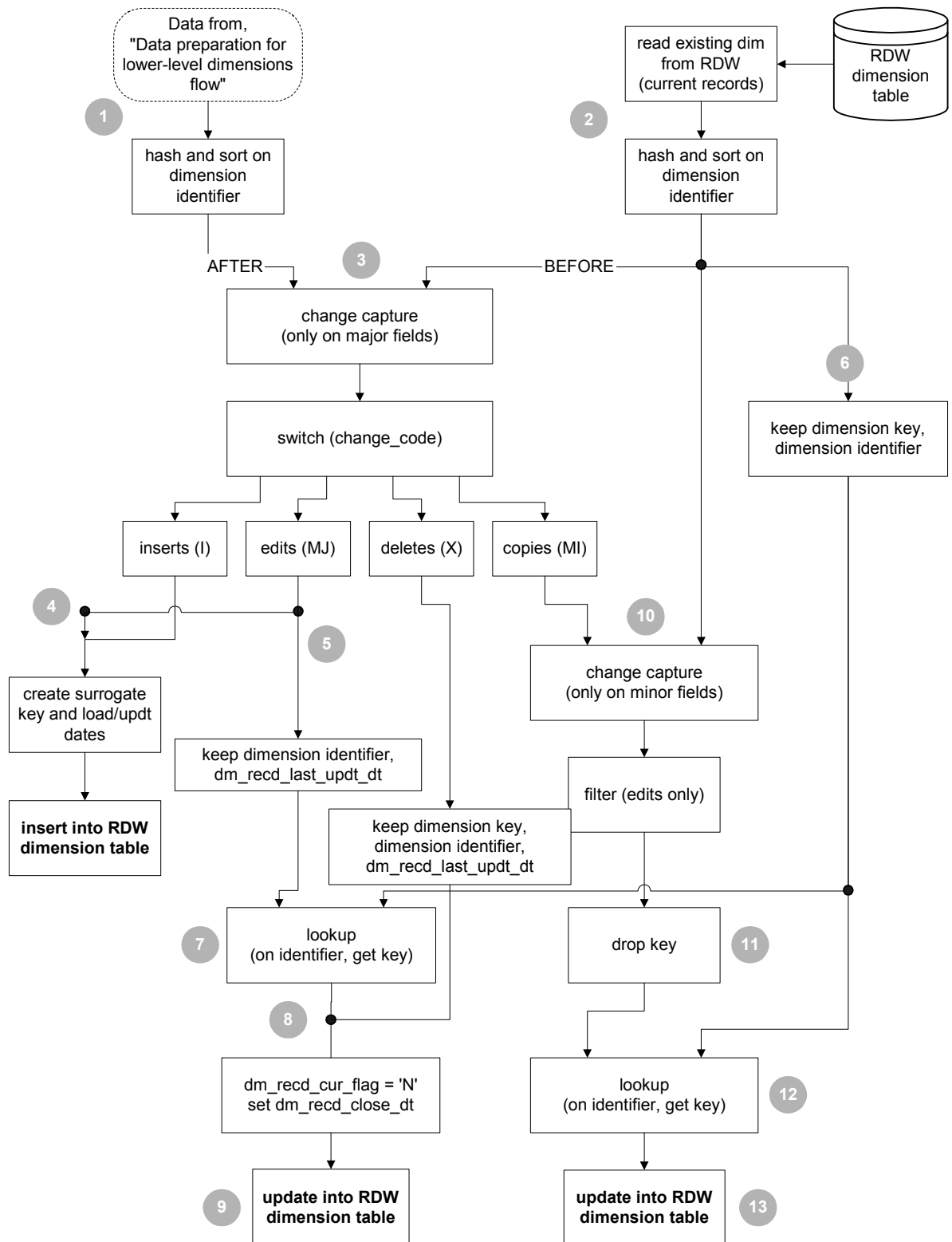
- 1 The current dimension data file is extracted from the source system (using RETL components). This file is transferred to RDW and loaded into an RETL dataset using an IMPORT operator and predefined schema file.
- 2 The GENERATOR operator adds the following maintenance columns to the dataset (see the section, “Maintenance columns in the DM table”, earlier in this chapter):
 - dm_recd_load_dt
 - dm_recd_last_updt_dt
 - dm_recd_close_dt
 - dm_recd_curr_flag

Note that although the columns are added to the dataset, the processing that occurs within the “Major and minor change capture for lower-level dimensions flow” described later in this chapter determines which columns are actually kept. In addition, a blank or default surrogate key is added, to enable the schema to match the target table in RDW.

- 3 The dataset is joined with one of the immediate rollup tables from RDW. For example, the LOC dimension dataset is joined with the ORG_DISTT_DM table to get the surrogate keys for district and all the dimensions above district, because these keys are redundantly stored in RDW.
- 4 This join with the dimension table above is repeated for every immediate rollup of a dimension. Thus, in the example above, region is not used for one of these joins, because it is not an immediate rollup of location, but is a rollup of district. However, when processing the item dimension, subclass would be joined to incoming item data, because it is an immediate rollup of item.
- 5 The final data is then the input for the next dataflow diagram, “Major and minor change capture for lower-level dimensions data flow”.

Major and minor change capture for lower-level dimensions flow

The diagram in this section describes the general RDW major and minor change capture for lower-level dimensions flow. Explanations of each numbered item on the diagram follow it.



Major and minor change capture for lower-level dimensions flow

Major and minor change capture for lower-level dimensions flow description

Note that the following numbers correspond to the numbers shown in the flow diagram above:

- 1 Data from the source system is already transformed to match the existing dimension table from RDW in all respects except that the surrogate key for the current dimension is not available (set to zero). Although the `dm_recd_load_dt` and other dimension maintenance columns are in the schema, whether each one is kept depends upon the type of processing to occur (for example, insert, edit, delete, and so on).
- 2 The data is read from the RDW table that stores the current dimension's information, filtered to contain only the current records (rows where `dm_recd_curr_flag="Y"`).
- 3 The CHANGECAPTURE operator compares the two incoming datasets and adds a 'change_code' field to the output, which indicates one of the following:
 - inserts (a record exists in the AFTER dataset, but not in the BEFORE)
 - deletes (the record does not exist in the AFTER dataset, but does exist in the BEFORE)
 - edits (a record exists in both datasets but with different values)
 - copies (a record exists in both datasets, and all the minor changeable fields are the same)

This CHANGECAPTURE operator looks only at fields that would cause a 'major' change, and ignores all other fields for the sake of comparison. The delete stream passes the records from the BEFORE dataset, whereas all other streams pass the AFTER dataset unchanged.

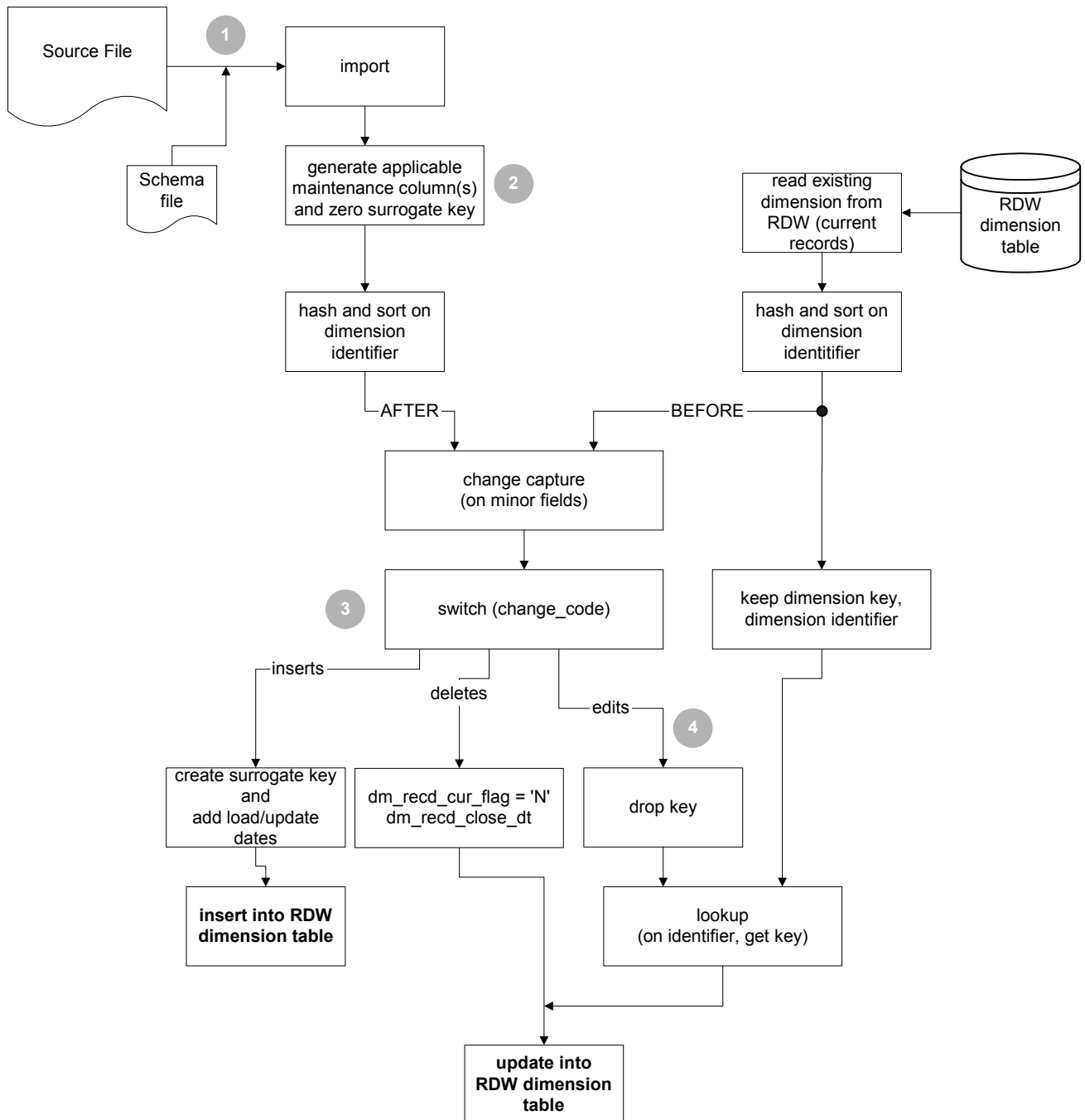
- 4 Inserts and edits have a new surrogate key generated for them, have the load and update dates set to the current date, and are inserted into the RDW dimension table. Because the incoming dataset from step one (1) has all the information necessary to fill in the RDW dimension table, these records can be directly inserted with no further joins. Edits result in an insertion and an update because the CHANGECAPTURE operator is detecting major changes, which result in the creation of a new RDW record, and the closing out of the old record.
- 5 Because major changes require closing out the old record, the edit stream also goes to a part of the flow that closes out old records. Closing out a record involves changing the value of the `dm_recd_curr_flag`, `dm_recd_close_dt`, and `dm_recd_last_updt_dt` fields, but no other fields. Because the CHANGECAPTURE operator passes all the fields from the AFTER dataset, all fields are removed except the `idnt` field, which is used to get the old surrogate keys. Thus, the resulting schema of the stream that is used to update RDW only contains fields that are to be updated, and the key. ('Idnt' is also there but is guaranteed to be the same because the compare is for the same idnts).

- 6 The RDW dataset is stripped to only the dimension idnts and keys, which are used as a lookup table to reattach the surrogate keys to datasets downstream. This step is intended to avoid field name conflicts and to stop downstream datasets from getting old values when undesired.
- 7 This lookup gets the old surrogate key for the current dimension for all updated/inserted records.
- 8 Because the remaining steps are to set the dm_recd_curr_flag to “N” and update dm_recd_last_updt_dt and dm_recd_close_dt, deletes and edits, at this point, can be considered together. The same set of fields—only the fields that are necessary to update the records—has been preserved for the delete stream.
- 9 The data is updated in RDW. Because RETL cannot update directly, this step involves a separate process. (See the passage, “Process to update records in RDW”, in Chapter 1.)
- 10 Records deemed as copies imply that no major change has occurred. However, it is possible for a minor change to have occurred. To prevent updating records where no change at all has occurred, this step compares the records again with the current RDW dataset, but this time, the comparison is executed on all minor fields. Only the records considered edits (that is, minor changes) are further processed.
- 11 The surrogate key field is dropped, to allow the actual surrogate key to be re-fetched in the next step (because the major CHANGECAPTURE operator will have lost the surrogate keys).
- 12 The original surrogate key is re-fetched using a join on the idnt field.
- 13 This stream of data is updated into the RDW tables using the standard process for updating records. (See the passage, “Process to update records in RDW” in Chapter 1.) Note that this stream contains many more fields than in step (9), because we effect minor changes. This logic implies that these streams cannot be combined.

Processing for the top-level dimensions

The flow diagram in this section—“Top-level processing data flow”—describes the processing of the highest levels in each dimension hierarchy. That is, this section addresses standalone non-hierarchical dimensions, such as currency, along with the highest level of a dimension hierarchy, such as a promotion event. None of the lookups described in the lower-level dimension processing section are required; thus, they do not appear in this flow.

This process has a very simple flow that imports data from the extracted file, compares it to the target data on minor changeable fields, and uses only the insert and minor change portions of the core change compare flow. The diagram on the next page shows this flow. Explanations of each numbered item on the diagram appear following the diagram.



Top-level processing data flow

Top-level processing data flow description

- 1 The current dimension data file is extracted from the source system (using RETL components). This file is transferred to RDW and loaded into an RETL dataset using an IMPORT operator and predefined schema file.
- 2 The GENERATOR operator adds the following maintenance columns to the dataset (see the section, “Maintenance columns in the DM table”, earlier in this chapter):
 - dm_recd_load_dt
 - dm_recd_last_updt_dt
 - dm_recd_close_dt
 - dm_recd_curr_flag

In addition, a blank or default surrogate key is added, to enable the schema to match the target table in RDW.

- 3 The CHANGECAPTURE operator in this case only compares against minor fields, because there are no major fields. Copies are discarded immediately.
- 4 Because changes can only be minor, there is no need to close out records. The lookup to reattach the old surrogate keys is still needed, but these records are then updated directly.

Datamart table

The datamart (DM) table is the final resting ground for dimensional entities. DM tables are visible from the front end. These tables are also used by fact loading modules to perform the following:

- Map identifiers to keys, which are then inserted into fact datamart tables.
- Determine hierarchical relationships for aggregation.

Note that these tables cannot be purged, unless the client wishes to manually roll-off or delete closed dimensional rows (for an item which no longer needs to be queried, for instance). Retek does *not* recommend that clients attempt such dimension data purging, and Retek provides no dimension purging code.

The table and the accompanying descriptions of the maintenance columns shown below illustrate how a record that reflects a change type is reflected in a DM table.

Dimension datamart (DM) table

	dm_recd_last_updt_dt	dm_recd_load_dt	dm_recd_close_dt	dm_recd_curr_flag
Inserted	Current processing date	Current processing date	4444-04-04	Y
Minor Changed	Current processing date	Original load date	4444-04-04	Y
Closed	Current processing date	Original load date	Current processing date	N
Major Changed Closed	Current processing date	Original load date	Current processing date	N
Major Changed Inserted	Current processing date	Current processing date+1	4444-04-04	Y

Chapter 3 – Fact data concepts

This chapter describes the following fact data concepts in RDW 10.2:

- An overview of RDW fact processing
- Fact functional areas
- Types of fact tables
- How fact data is extracted from the source system and loaded into RDW
- General fact processing
- Detailed fact load processing
- Fact aggregation processing
- Fact matrix table processing

An overview of RDW fact processing

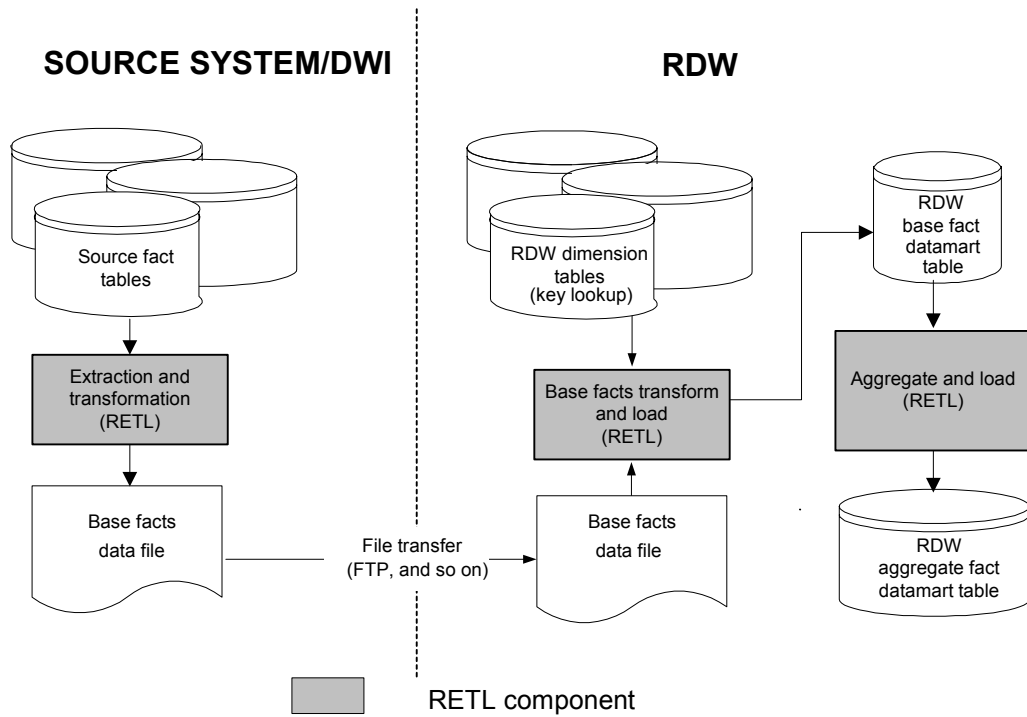
The following description and “Overview of fact, extraction, load, and aggregation” diagram offer an overview of RDW’s fact process.

Fact data is extracted from the source system using RDW’s data warehouse interface (DWI). The DWI is RETL code that extracts the data and writes it to a text file. Only the changed and/or new facts for today are extracted. Note that in lieu of the extraction, a client could generate a source flat file.

The data file is transferred to the RDW server using a common data transfer process such as FTP. In RDW, the data is pulled from the data file, and identifiers are mapped to the appropriate surrogate keys. (See Chapter 2, “Dimension data concepts”, for a discussion of surrogate keys.)

RDW base fact processing modules, which are Korn shell scripts containing RETL operators and RDW library calls, transform and load data to base fact datamart tables. The next step is the aggregation process, where the data is read from one base fact datamart table, aggregated, and then loaded into one fact aggregate datamart table.

As with dimension processing, see Chapter 8, “Program reference lists”, for module-specific information, such as PROGRAM_CONTROL_DM values, command-line parameters, and so on.



Overview of fact extraction, load, and aggregation

Fact functional areas

Fact data represent transaction values extracted from a source system such as Retek Merchandising System. RDW's fact functional areas are listed in the following table:

RDW 10.2 Fact Functional Areas	
Competitor Pricing	Cost
Exchange Rates	Inventory Adjustments
Inventory Position	Inventory Receipts
Inventory Transfers	Loss Prevention
Sales Markdowns	Market Sales Data
Net Cost and Profit on Base Cost	Pack Sales
Planning (TopPlan)	Pricing
Inventory Return to Vendor	Sales Forecasts
Sales Productivity	Sales Transactions
Space Allocation	Stock Ledger
Store Traffic	Supplier Availability
Supplier Compliance	Supplier Contract
Supplier Invoice Cost	Transaction Tender
Unavailable Inventory	Vouchers

Fact table types: base and aggregate

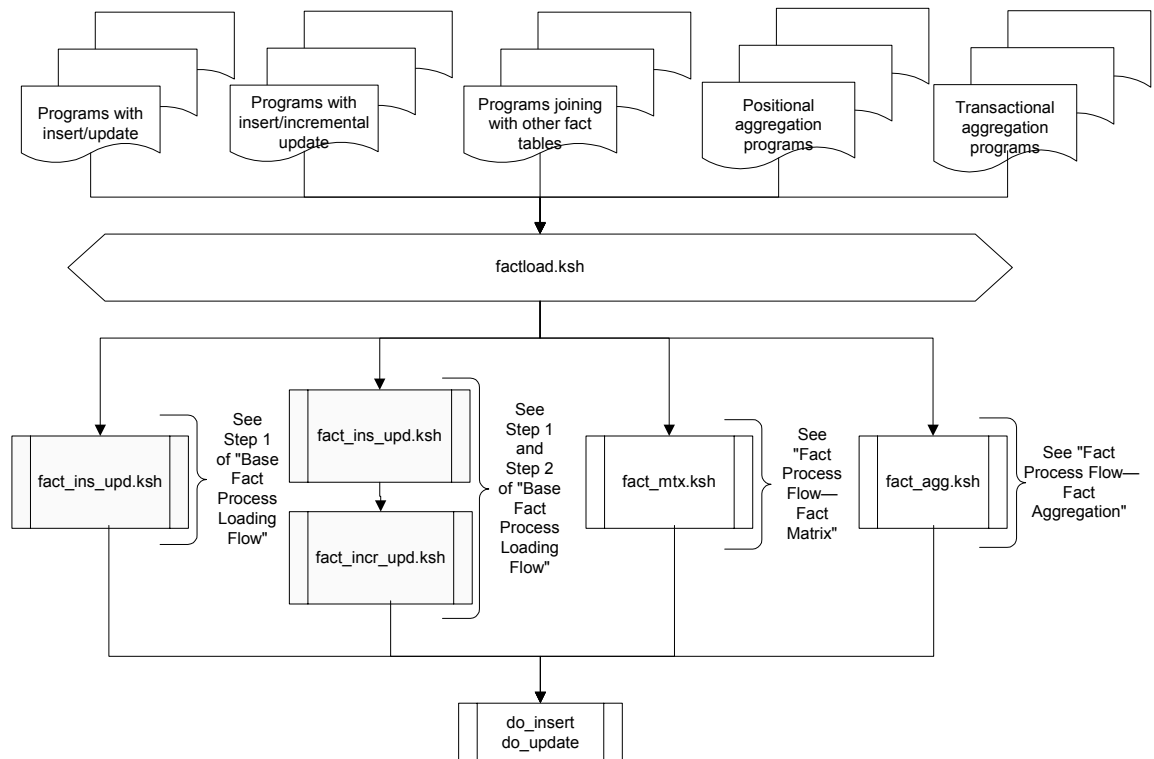
RDW contains two types of tables: base and aggregate.

- A 'base' fact table holds fact data for a given functional area at the lowest level of granularity. The process of populating a base fact table begins with the RETL extraction of the data from the source system. The extraction results in a text file that is sent to RDW. In RDW, a RETL transformation and load process accepts the fact data file and updates the base table. In order to use RETL to load data files, the RDW fact API defines a schema file to describe the target table columns and data types for each base fact datamart table. RETL references the schema for loading source data files. Data on the base fact table is then aggregated.
- A fact 'aggregate' table holds fact data rolled up from the base table to a higher level of a dimensional hierarchy. RDW uses Korn shell scripts and RETL operators in order to aggregate data.
- Non-compressed fact data can be purged or rolled off whenever a client no longer wishes to query the data. Retek provides no purging routines because purging must be determined by client-specific business requirements. For more information concerning compressed fact tables, see Chapter 4.

General fact processing

The following diagram illustrates the fact process flow in RDW 10.2. The flow proceeds from the fact programs (that require the use of sub-libraries) to the factload.ksh. This library interprets the needs of the programs in order to direct them to call the correct sub-library or sub-libraries. Factload.ksh thus plays the role of the ‘library traffic cop’. Note that almost every fact program that uses sub-libraries must call factload.ksh so that it can be properly directed. Once the applicable sub-library has processed the program, the system can make the correct changes to RDW’s fact tables. The very few standalone modules that do not use factload.ksh are not shown in the diagram.

The flow diagrams described later in this chapter illustrate specifically how and in what context data is processed within each applicable Korn shell sub-library. Thus, adjacent to each sub-library in the diagram below is a callout that refers to the specific process flow diagrams (and any applicable steps therein) that are described later in this chapter.



Fact process flow—general

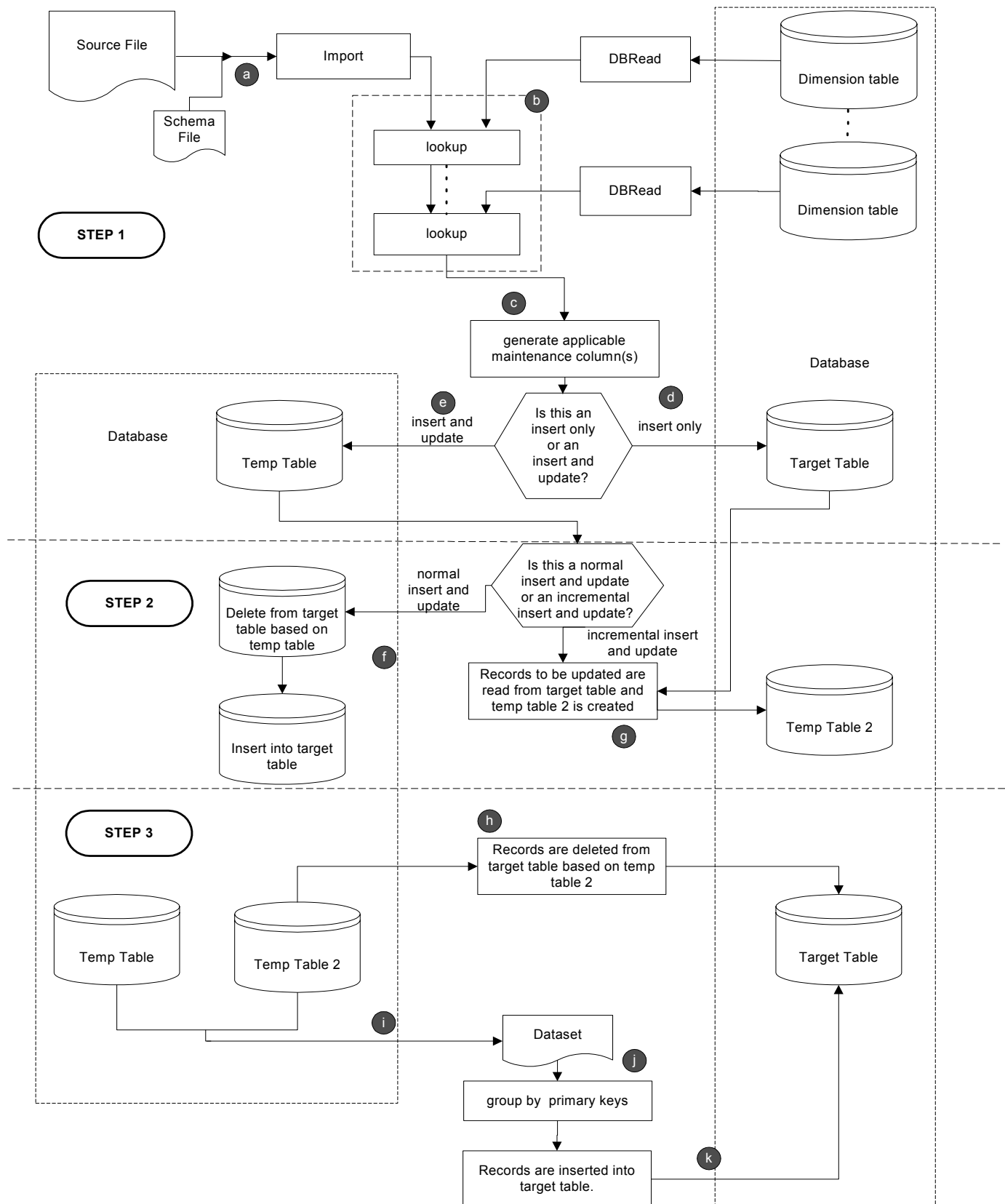
Detailed fact load description

This section describes the process of transforming and loading fact data. In order to use RETL to load data files, RDW uses a schema file to describe the source file fields and data types for each base fact datamart table.

The following diagram describes the general RETL fact process and represents most of the fact loading processes. However, note that the following issues are treated differently for each individual program:

- Some base facts use different dimension tables to do lookups.
- Some base fact loadings only have INSERTs with no UPDATEs (compressed datamarts, for example).
- Some base facts (such as merchandise sales) have incremental updates where the new value is a summation of an old and a new value.

Explanations of the numbered steps and lettered items follow the diagram.



Base fact loading process flow

Base fact loading process flow description

Step 1

- a The fact data file extracted from the source system is loaded into a RETL dataset using the IMPORT operator, based on the schema file that defines all the data fields and data types in the data file that, in turn, are based on the target table.
- b The DBREAD operator is used to read all joining dimension tables into RETL datasets as lookup tables for incoming data in order to get keys based on the identifiers. The number of dimension tables varies for each fact program. Module dimlkup.ksh generates the RETL code (including the DBREAD operator) that selects the data from the dimension tables and joins it with the incoming data. Should the need arise to customize this process, the client can change the variables within dimlkup.ksh.
- c A maintenance column is generated that acts as a date marker (that is, a time stamp). Essentially, this maintenance column records that fact that these rows have been altered on this day.
- d For insert-only fact programs (for example, exchange rate, cost, and so on), the resulting dataset can be appended into the target table directly. For these programs, this step is the end of the process.
- e For fact programs with insert and update records (for example, space allocation, net cost, and so on), the dataset (containing the new records) is written into a temporary table (base temp table).

Step 2

- f For all base fact programs with normal insert and updates, this temporary table is used to determine which of the old update records in the target table should be deleted. The old records are deleted from the target table. The new records are inserted into the target table. For these programs, this step signifies the end of the process.
- g For all base fact programs with incremental insert and updates, the records to be updated are read from the target table and a second temporary table (temporary table 2) is created.

Step 3

- h The temporary table 2 is used to determine which of the old update records in the target table should be deleted, and those records are deleted.
- i The records in the temporary table and in temporary table 2 are combined to form a new dataset.
- j The new dataset is grouped by the primary keys of the target table to sum up the required fact fields.
- k The resulting dataset is written to the target table (that is, the records are inserted into the target table). For base fact programs with incremental insert and updates, this step signifies the end of the process.

Fact aggregation

After facts are loaded into the base datamart tables, the process of aggregation begins. Aggregation refers to the process of taking data at a particular level of granularity, for example the item level, and summing it up to a higher level, such as the subclass level, in order to improve query performance. In order for the front end to accurately drill between levels, the names of fact columns must remain the same between the base level and all aggregate levels.

There are two primary types of aggregation in RDW: positional fact aggregation and standard fact aggregation. Positional aggregation updates a value to the current amount at the current time. Standard aggregation sums up all values to the current time. A third aggregation type called a ‘derived datamart’ also exists that supports some complex metrics.

Positional fact aggregation

Some fact tables in the RDW contain information about an entity’s position or status at a given point in time. Such data does not sum up in the same way that transactional data does. See the section “Standard fact aggregation” later in this chapter. For instance, the pricing datamart contains unit retail values for a given item at a given location. Even though new records are written to the table only when a price changes, a user must be able to query for any day and have the system return the correct value. However, storing positions for every item at every location for every day quickly becomes prohibitive from a data storage and load performance standpoint. In order to strike a balance between storage and performance, RDW makes use of a technique called compression to store and report on positional facts. See Chapter 4, “Compression and partitioning”, for more information about how compression works and where RDW uses it.

RDW 10.2 contains four positional fact aggregation modules. They are listed in the table below.

Positional Fact Aggregation Modules	
Invilwdm	Compressed source and target table
Invblddm	Non-compressed source (cur table) and target table
Invblwdm	Non-compressed source and target table
Sfcblwdm	Non-compressed source and target table

Positional fact aggregation over time

Because data on positional fact tables reports on the state of an entity at a certain point in time, rather than the total activity of an entity, these facts cannot be simply summed over time. For instance, the question: “What was my total unit retail for this week?” is nonsensical. For this reason, aggregations of positional facts along the axis of time take end-of-period snapshots that answer the question: “What was my unit retail at the end of this week?”

With all aggregations along the time axis, aggregation programs run daily. For aggregations of positional facts within a period, this results in a period-to-date position, rather than an end-of-period position. Once the period is complete, the last run of that period results in the desired end-of-period position.

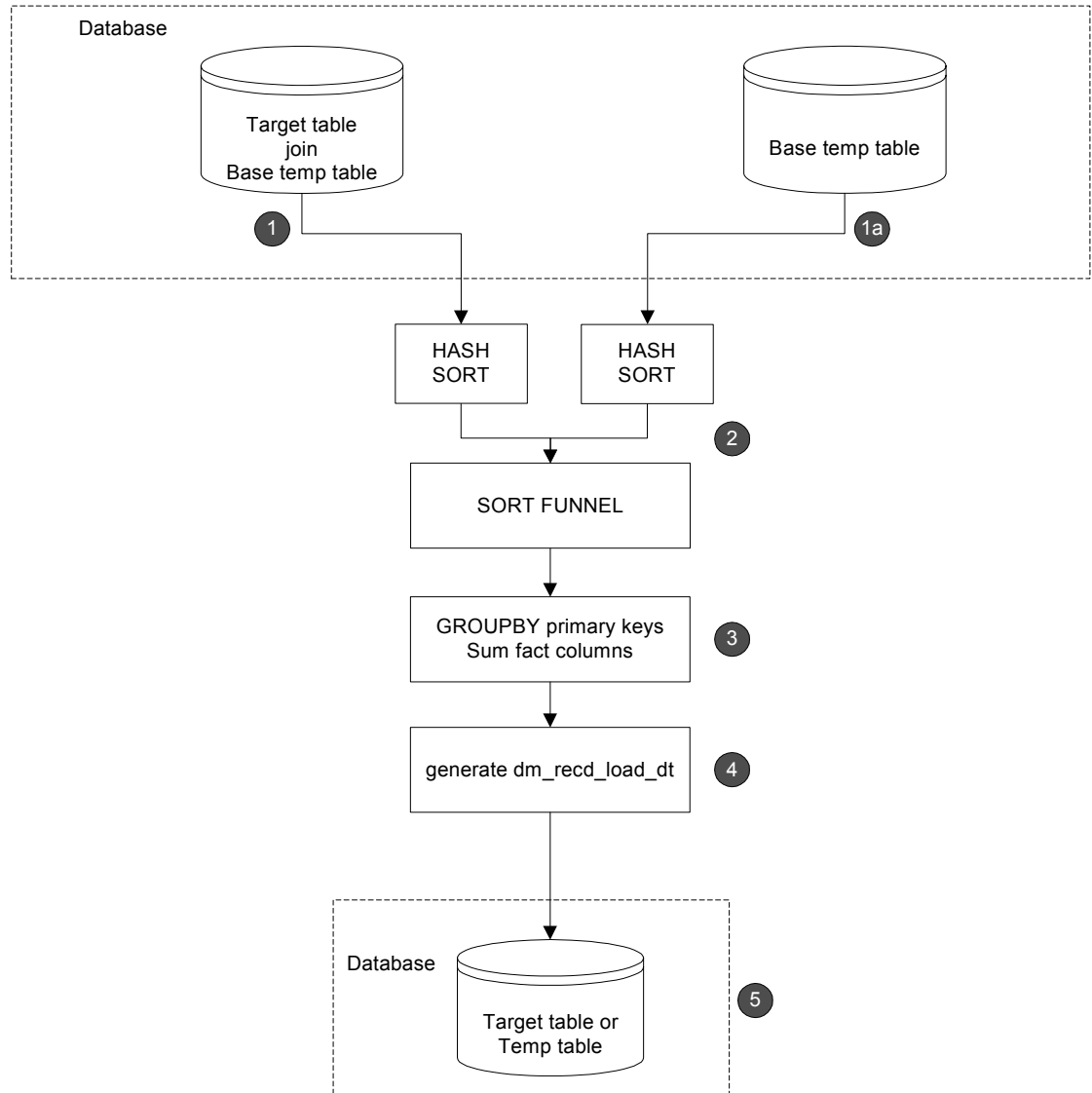
Decompressed aggregates

The compression of positional facts is complex. In order to simplify maintenance and to maximize performance, it is sometimes better to leave base-level facts in their raw compressed state, and to store higher-level aggregates (with less fine levels of granularity) in a decompressed state, in which positions for all entities are written everyday. Building these decompressed aggregates can be a significant task in itself because it involves finding the current positions for every entity at the lower level for the current point in time—even for those entities that may have last had a record some time ago. Fortunately, this task can be simplified by the use of a current position table (such as `INV_IL_CUR_DM`). A current position table is used, for example, when facts are aggregated from item-location-day to subclass-item-location-day. Less frequently, loads may also make use of a temporary table, which only contains today’s changes, to facilitate bulk processing of the data. For example, when facts are aggregated from item-location-day to item-location-week, the aggregation does not include the entire week’s data, only today’s changes.

Standard fact aggregation

Most fact tables in RDW contain information about some sort of activity, or transaction, that has occurred. For instance, the merchandise sales tables contain total sales values for a given item at a given location on a given day. This is the simplest type of fact data in RDW. All the data is there and can be summed up along any dimensional axis for reporting purposes.

The following diagram shows the standard fact aggregation process. Explanations of the numbered items follow the diagram.



Fact process flow—fact aggregation

Fact process flow—fact aggregation description

- 1 The target table is joined with the base temp table in order to select the rows from the target aggregate table that need to be re-aggregated because data has been changed on and/or inserted to the base temp table today.

For example:

Target Aggregate Table			
Location	Item	Week	Amount
A	B	1	20
A	C	1	30

is joined with:

Base Temp Table (today)				
Day	Location	Item	Week	Amount
4	A	B	1	5
4	A	D	1	70
4	A	F	1	30

to produce:

Rows that Need to be Re-aggregated			
Location	Item	Week	Amount
A	B	1	20

- 1a The base temp table is the collection of changed and new data that needs to be re-aggregated.

For example:

Base Temp Table			
Location	Item	Week	Amount
A	B	1	5
A	D	1	70
A	F	1	30

- 2 Each RETL dataset is hashed and sorted (by the HASH and SORT operators) in the order of its primary key. Based on the primary key of each, the SORT and FUNNEL operators combine the datasets into one that is sorted.

For example:

RETL Dataset after SORT and FUNNEL			
Location	Item	Week	Amount
A	B	1	20
A	B	1	5
A	D	1	70
A	F	1	30

- 3 Aggregation takes place on the primary key because of the work of the GROUPBY operator, which facilitates the summation of the fact columns.

For example:

RETL Dataset after Aggregation			
Location	Item	Week	Amount
A	B	1	25
A	D	1	70
A	F	1	30

- 4 A maintenance column is generated that acts as a date marker (that is, a time stamp). Essentially, this maintenance column records the fact that these rows have been altered on this day.
- 5 The data is written to either:
- the target table if the applicable programs contained only inserts.
 - a temp table if the applicable programs contained updates. The target table then undergoes a normal update process. If necessary, see the passage, “Process to update records in RDW”, in Chapter 1.

Derived datamarts

To support some complex metrics, it is sometimes necessary to build an aggregate table with facts that are more than simple sums of those lower levels of granularity. This is similar to standard aggregation in that data moves from one fact datamart table to another. However, because the fact column names are different, there is no straight drill path between the two levels. As a result, these higher-level DM tables are not really aggregates in the purest sense; rather, they are different datamarts derived from a lower level. Here is an example.

The Sales datamart contains profit calculated using an item's weighted average unit cost. The Net Cost datamart holds various costs for an item, from a given supplier, that are used for a more detailed profit analysis. By combining data from these two functional areas—the Net Profit datamart is built. By deriving a datamart, the user can view profit analysis reports in the front end without the use of overly complex metrics. An additional benefit of deriving a datamart is that database performance improves.

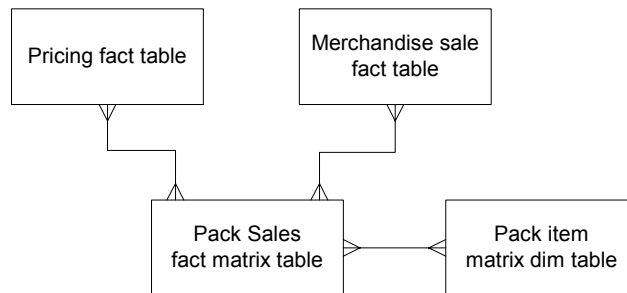
Derived datamarts in RDW 10.2 include the following:

- Sales Transaction Summary
- Tender Transaction Summary
- Loss Prevention Transaction Summary
- Supplier Compliance Summary
- Net Profit
- Pack Sales / Pack Sales Markdowns
- Voucher Movement

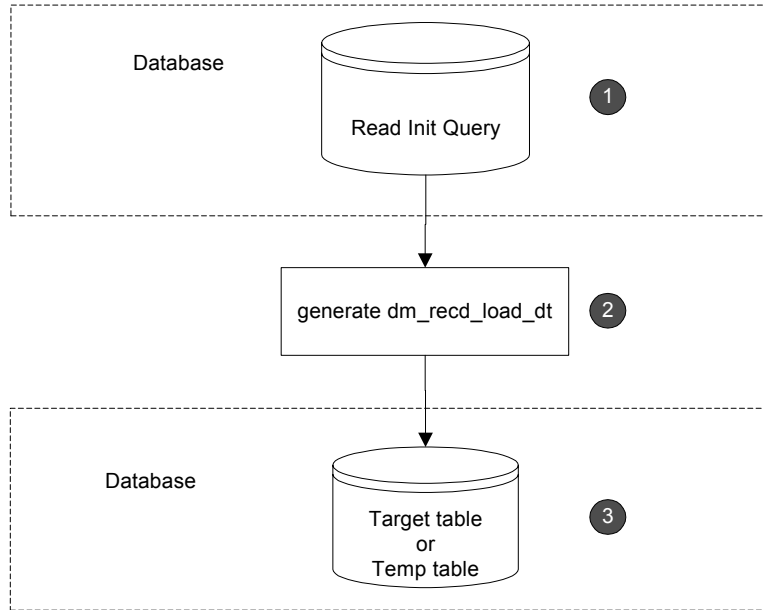
Fact matrix processing

A derived datamart can also be thought of as a ‘fact matrix’. As the diagram and table below illustrate, the matrix table, by having the same dimension key, resolves the relationship among fact and dimension tables that have, in terms of their cardinality, a many-to-many relationship.

For example:



Fact Matrix Table Example				
	Location	Day	Cost Amount	Sale Amount
1	1	2	5	10
1	2	1	10	20
2	1	1	5	10



Fact process flow—fact matrix

- 1 When the program calls the library (fact_mtx.ksh), the RETL database read operator (for example, ORAREAD), within a query, retrieves the data. The query can, if necessary, perform aggregation.
- 2 A maintenance column is generated that acts as a date marker (that is, a time stamp). Essentially, this maintenance column records the fact that these rows have been altered on this day.
- 3 The data is written to either:
 - the target fact matrix table, if the applicable programs contained only inserts.
 - a temp table if the applicable programs contained updates. The target fact matrix table then undergoes a normal update process. If necessary, see the passage, “Process to update records in RDW”, in Chapter 1.

Chapter 4 – Compression and partitioning

This chapter describes how RDW 10.2 implements compression, and offers a discussion of Oracle partitioning.

Overview of compression

Although data warehouses are often very large, the amount of detail generated in some RDW tables is enormous even by usual standards. For example, a retailer with 500,000 items and 500 locations would generate 250,000,000 new rows each day. Storing this amount of uncompressed data is impractical from a disk storage perspective, both in the cost to store the rows and in the cost to perform backups and other database maintenance operations.

One approach that RDW uses to reduce the data volume is compression. This chapter describes:

- What compression does
- The mechanics of compression
- Which tables are currently compressed
- The Oracle features that are related to compression
- Strategies for implementing compressed tables for Oracle clients

What compression does

Compression refers to storing physical data that only reflects changes to the underlying data source, and filling in the gaps between actual data records through the use of database views. This method is engaged primarily for subject areas that are perpetual, such as inventory. For example, when querying sales data, a valid sale record either exists (a sale occurred) or a record does not exist (no sale occurred). However, when querying for on-hand inventory, even if no change occurred to the inventory on the date desired, a valid value is still required. One way to resolve this discrepancy is to store a record for every day for every valid item-location combination as mentioned above. Another method, compression, allows for the storage of only changes to the inventory position. The query is resolved by looking backward through time from the desired date (if no change record exists on that date) until an actual change record is found. This method returns the correct current data with the minimum requirements necessary for processing and storing data.

The mechanics of compression

The purpose of decompression views is to give the application the illusion that there is a record for each possible combination (for example, an item-location-day record for each permutation) when in fact there is not. Thus, the fact of whether a table is compressed or not should not be visible to the application that queries data from that table.

A compressed table is made up of two distinct parts: a ‘seed’ that consists of all existing combinations at a point in time (typically the first day or week of the table or partition) and the changed data since that time.

When resolving a query for a particular record, the decompression view provides the latest record for the requested item and location with the maximum day that is less than or equal to the requested day. A decompression view needs to encompass both the seed and all of the changed data since that seed.

To illustrate how the decompression views actually work, assume the following: I am interested in the inventory position of item 10 at location 10 on 1/23/02. The seed was done on 1/1/02. Changes were posted on 1/4/02, 1/15/02, and 1/30/02. The row that is presented to the application by the decompression view is the row on 1/15/02, because it is the greatest date that is less than or equal to the requested date. As a second example, assume that the inventory position of item 10, location 10, day 1/3/02 was desired. Because there was no change record less than or equal to the desired date, the seed record from 1/1/02 would be presented to the application.

Compression’s performance is excellent when the user is querying for a single day (as in the example above). When querying over a group of days, however (for example, all of the inventory positions at a given location on a given day), the performance can be unacceptable. Even though the user is requesting a group of information back and in most cases the database can process groups of information efficiently, each individual row must be evaluated individually by the decompression view and cannot be processed as a group. To counteract the slow performance of these summary operations, Oracle clients may take advantage of compressed table partition seeding (see the passage, “Partitioning for the Oracle client only”, later in this chapter).

This partition seeding utilizes CUR tables (also known as ‘current’ tables). An example is the INV_IL_CUR_DM table, which holds the current decompressed position for every item and location on the INV_ITEM_LD_DM table. This position can be used (by Oracle clients only) as a partition seed. This position is also utilized by base RDW code during major change fact seeding (see the passage, “factopendm.ksh”, later in this chapter).

Compressed tables and CUR tables

The table below illustrates the compressed tables within RDW 10.2, along with their corresponding CUR tables.

Compressed Tables	CUR Tables
CMPTR_PRICING_ITEM_LD_DM	CMPTR_PRICING_IL_CUR_DM
COST_ITEM_SUPP_LD_DM	COST_ISL_CUR_DM
EXCHNG_RATE_CRNCY_DAY_DM	(No CUR table)
INV_ITEM_LD_DM	INV_IL_CUR_DM
INV_ITEM_LW_DM	(No CUR table)
INV_UNAVL_ITEM_LD_DM	INV_UNAVL_IL_CUR_DM
NET_COST_SUPP_ITEM_LD_DM	NET_COST_SIL_CUR_DM
PRICING_ITEM_LD_DM	PRICING_IL_CUR_DM
SPACE_ALLOC_DEPT_LD_DM	SPACE_ALLOC_DL_CUR_DM
SPACE_ALLOC_ITEM_LD_DM	SPACE_ALLOC_IL_CUR_DM
SUPP_AVAIL_ITEM_DAY_DM	SUPP_AVAIL_ITEM_CUR_DM
SUPP_CNTRCT_ITEM_DAY_DM	SUPP_CNTRCT_I_CUR_DM

Coping with major changes

Factclosedm.ksh

On a compressed fact table, a record is only posted to the table when there is a change in one of the fact attributes. If there is no activity, no record is posted. Decompression views then fill in the gaps between physically posted records so the front end can see a fact record for each item-location-day combination. However, when an item or location or department is closed or major-changed, any fact record with those dimensions becomes inactive. The decompression views need to be informed to stop filling in the gap after the last record was posted. To accomplish this instruction, factclosedm first queries the PROD_ITEM_RECLASS_DM, ORG_LOC_RECLASS_DM, and PROD_DEPT_RECLASS_DM tables (see the section, “factopendm.ksh”, later in this chapter) to determine what compressed item-location facts need to be closed today. Factclosedm then inserts a ‘stop record’ that has a DM_REC_STATUS_CDE = ‘X’. The decompression view fills in records up to the day that a status code of ‘X’ is posted. The close record is inserted for tomorrow’s DAY_IDNT, indicating that the fact record is no longer valid beginning tomorrow, when the newly seeded record (from factopendm.ksh) becomes active. In the case of the one compressed week table, INV_ITEM_LW_DM, factclosedm inserts close records for next week’s WK_IDNT.

Factopendm.ksh

RDW Data Compression tables require seeding when a major change in the product and/or organization dimension causes new surrogate keys to be created for items or locations. Seeding the compressed tables is required because the new key represents a new hierarchy relationship. If the new key is not represented on the compressed table, the compression view does not pick up any data from the day the old dimensions were closed to the day a record with the new dimensions is posted to the compressed fact tables. This missed data causes inaccuracy in query results and incorrect data aggregation.

There are two steps to this seeding process. First, the modules prditmdm.ksh, prddepdm.ksh, and orglocdm.ksh run as part of the dimension loading process to populate temporary tables PROD_ITEM_RECLASS_DM, PROD_DEPT_RECLASS_DM, and ORG_LOC_RECLASS_DM. Next, the module factopendm.ksh looks at the tables for reclassified items, departments, or locations. It seeds all of the compressed tables with records that contain the reclassified ITEM_KEYS, DEPT_KEYS and/or LOC_KEYS.

Partitioning for the Oracle client only

Overview of partitioning strategies

Currently, RDW 10.2 base code ships with no tables partitioned. Because RDW 10.2 is database independent and partitioning only applies to Oracle clients, this section describes optional partitioning strategies for compressed datamarts that clients using Oracle may wish to pursue.

As described earlier, ‘decompression views’ provide a virtual view of a fully populated table even though the underlying table is actually only sparsely populated. For large compressed tables, especially the INV tables, splitting them into table partitions can provide the following benefits:

- Partitions are smaller and therefore easier to manage
- Management operations on multiple partitions can occur in parallel
- Partition maintenance operations (such as index rebuilds) are faster than full table operations
- Partition availability is higher than table availability (for example, if I am recovering a particular partition, users may access all other partitions of the table at the same time)
- The optimizer can prune queries to access data in only the partition of interest, not the entire table (for example, if I am interested only in February’s data, I do not need to look at any of the table’s data outside of the February partition)
- Partitions are separate database objects, and can be managed accordingly (for example, if December sales are frequently accessed throughout the year whereas other months are not, the December sales partition could be located in a special tablespace that allows for faster disk access)
- In some situations, Oracle can create parallel operations on partitions that it cannot on tables; an example is joining between two different tables if they are partitioned on the same key (this feature is called a ‘parallel partition-wise join’)

The general guideline is that table partitions should be used on very large tables. Tables larger than 20 GB would be potential candidates for partitioning. For optimal performance on inventory tables, partitioning is highly recommended.

Indexes, as well as tables, can be partitioned. ‘Index partitions’ can be global (one index over the table, regardless of whether the table is partitioned or not) or local (there is a 1-to-1 correspondence between index partitions and table partitions). In general, when tables are partitioned, local indexes should be preferred to global indexes for the following reasons:

- Maintenance operations involve only one index partition instead of the entire index (for example, if the oldest table partition is aged out, a local index partition can be dropped along with its corresponding table partition, whereas an entire global index would need to be rebuilt after it becomes unusable when a table partition is dropped)
- The optimizer can generate better query access plans that use only an individual partition
- When multiple index partitions are accessed, the optimizer may choose to use multiple parallel processes rather than just one

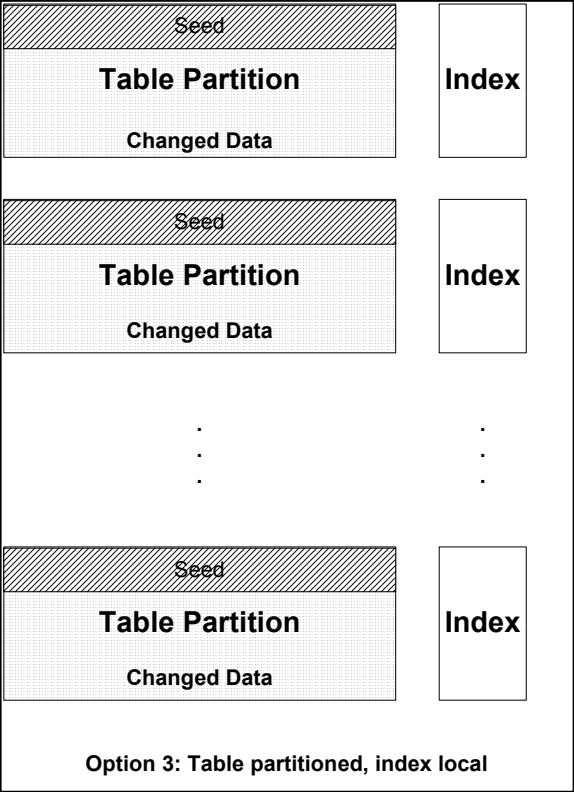
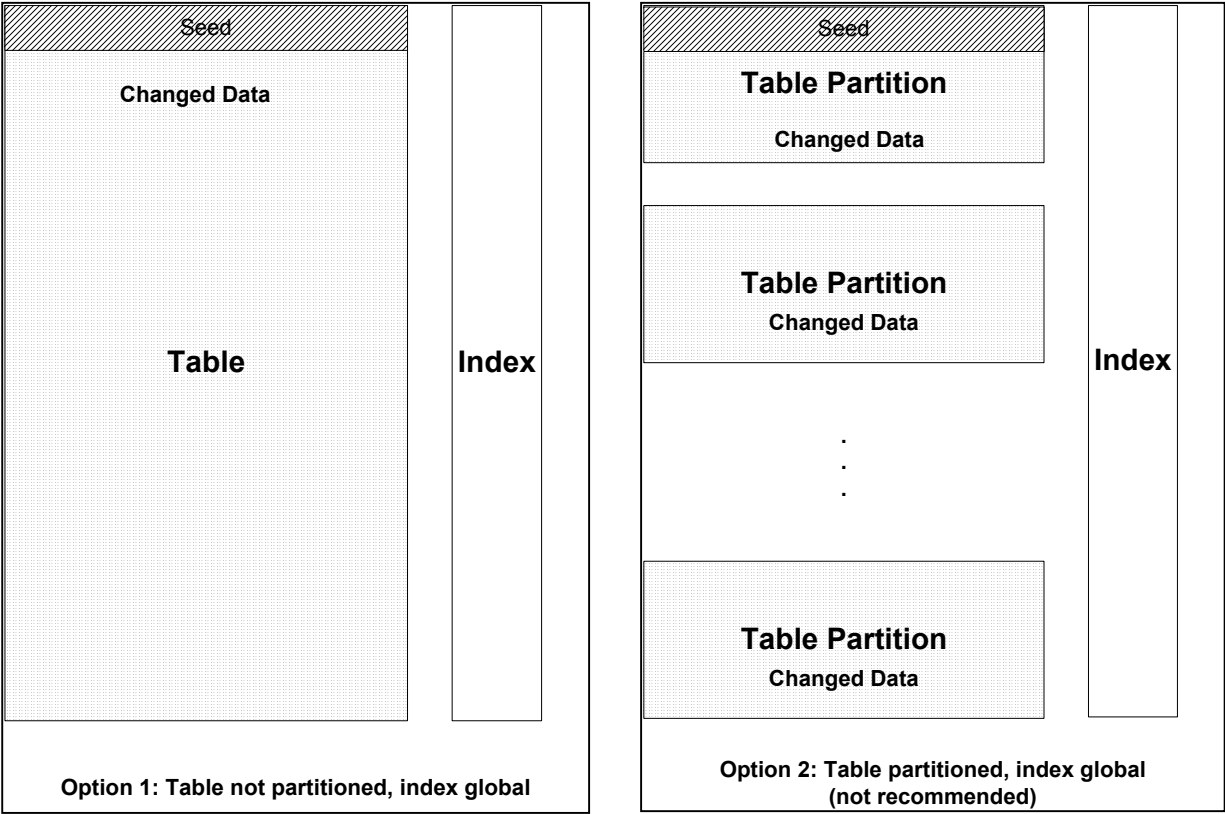
Implementing RDW partitioning

For those clients who choose to partition a compressed table, the diagrams on the following page illustrate some of the possibilities for table and index layout.

In general, option 3 is the preferred solution for large compressed tables (typically, the INV_ITEM_LD_DM and INV_ITEM_LW_DM tables). It uses table partitions and local indexes, thus minimizing the impact of index maintenance and the deletion of old table partitions.

Option 1 is viable for smaller compressed tables (typically, all other compressed tables besides INV). The disadvantage is that there is no way to delete historical data (that is, the table continues to grow).

Option 2 is not recommended. It combines the disadvantage of not allowing for historical data to be purged with the disadvantage of having a global index on a partitioned table.



Partitioning strategy and requirements for MicroStrategy 7

Partitioning allows a large fact table to be split into several smaller tables. The appropriate partitioning strategy can both significantly improve query response time and decrease the time required to load tables. These advantages must be weighed against the increased database maintenance that is required in a partitioned environment.

Partitioning can be effective for large tables that allow splitting along the time dimension. For example, a sales fact table might be partitioned by year in an environment in which the majority of queries retrieve data for the current year. In such a case, performance would increase because the majority of queries would be run against a smaller table. Although time is frequently used to partition tables, MicroStrategy 7 allows partitioning along any dimension.

The following two methods are available for partitioning in a MicroStrategy 7 environment:

- Warehouse partition mapping manages partitioning using mapping tables. These must be created and managed by the DBA.
- Metadata partition mapping manages partitioning from the MicroStrategy 7 metadata. This method eliminates much of the maintenance at the database level.

Note: For Oracle clients who implement partitioning for compressed tables such as INV_ITEM_LD_DM, MicroStrategy 7 *requires* the use of warehouse partitioning (PMT tables).

Warehouse partitioning

Warehouse partitioning requires a partition mapping table and the partition base table. These tables must be added to the MicroStrategy project.

Partition mapping table (PMT)

The DBA must create a table that maps the new tables according to the attribute used to create the partition. The PMT table must have the following structure:

ATTRIBUTE_ID	PBTNAME

The ATTRIBUTE_ID column contains the values of the attributes on which the table is partitioned. The PBTNAME (Partition Base Table Name) column contains the names of the partitions.

The PMT table for a partition by year would resemble the following:

YR_ID	PBTNAME
1997	Y1997_Sales
1998	Y1998_Sales
1999	Y1999_Sales
2000	Y2000_Sales
2001	Y2001_Sales

Multiple attributes may be used to create partitions. For example, you might partition by year and region. In this case, the PMT would contain the year and region identifiers and the corresponding PBT names.

Metadata partitioning

In this method, partitions are mapped in the project metadata, eliminating the need for the PMT. These objects, sometimes referred to as ‘data slices’ are filters that define the contents of the partition base tables. These objects are created with the Metadata Partition Mapping Editor.

An example of setup and maintenance for partitioning and warehouse partition mapping for compressed inventory tables

- 1 Make the following determinations, among others:
 - Your partitioning strategy
 - The time period your partitions will use
 - The 'values less than' boundaries according to your calendar
 - How many partitions are to be used
 - The partition naming standard
- 2 Create the partitioned tables and the partition mapping tables, in order to implement warehouse partitioning through MicroStrategy 7.
- 3 Verify you have populated the Time Calendar Dimension. See the RDW 10.2 Database Installation Guide for details.
- 4 Create partitioned decompression views for Inventory Position tables (that is, INV_ITEM_LD_DM and INV_ITEM_LW_DM) and other compressed tables you are planning to partition.
- 5 Re-run the standard grant and synonym scripts. See the RDW 10.2 Database Installation Guide for details.
- 6 Populate the Partition Mapping Tables PMT_INV_ITEM_LD_DM and PMT_INV_ITEM_LW_DM as well as any other compressed PMT tables that you have created.

Perform step numbers 4, 5, and 6 whenever any of the following events occur:

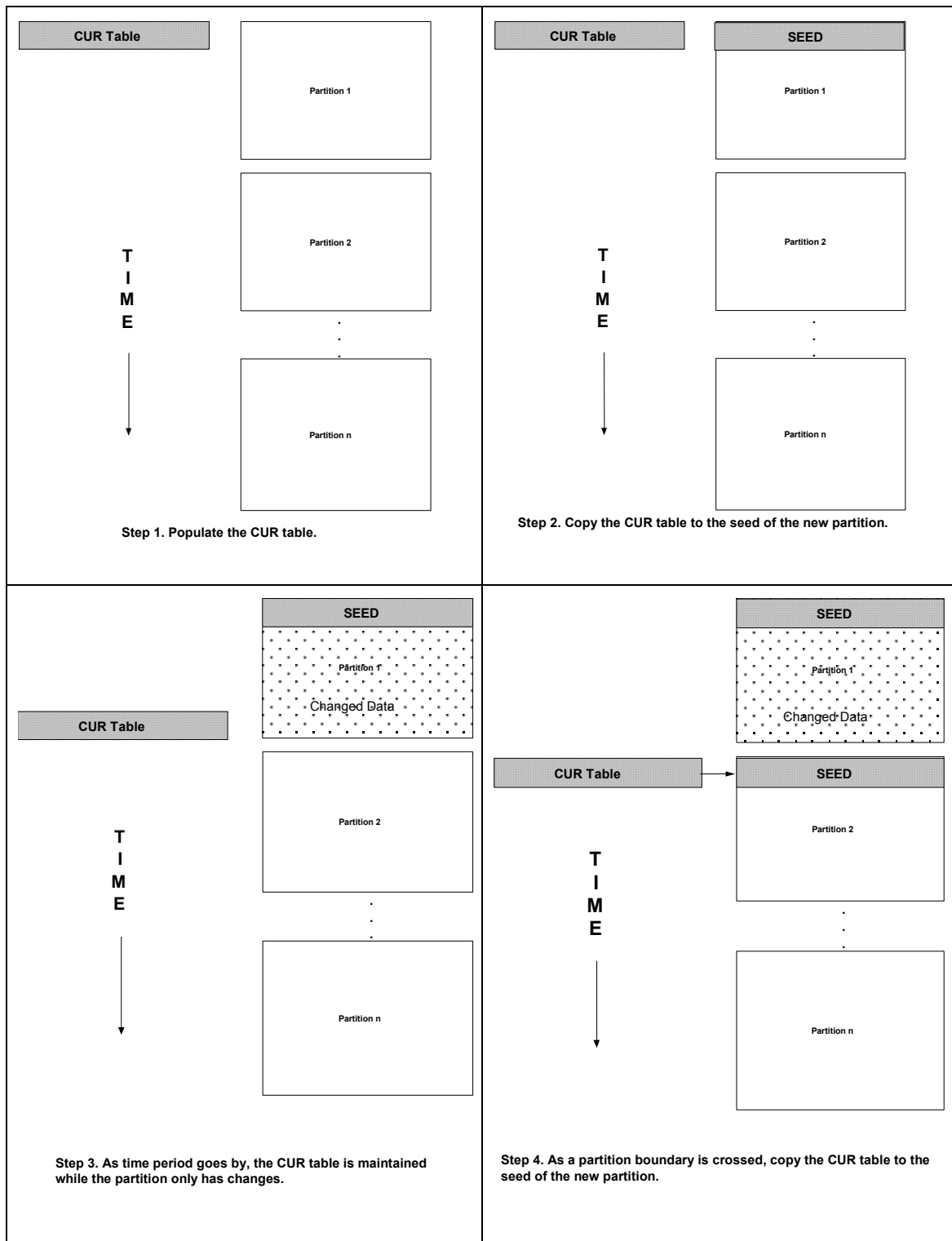
- Records are added to or deleted from the Time Calendar tables
TIME_DAY_DM or TIME_WK_DM
- Partitions are added to the Inventory Position tables
INV_ITEM_LD_DM or INV_ITEM_LW_DM
- Partitions are removed from the Inventory Position tables
INV_ITEM_LD_DM or INV_ITEM_LW_DM

Implementing partitioning for compressed inventory tables

Once the tables (including partitions) and indexes have been created, the data must be loaded. For tables that have a corresponding CUR table (such as INV_ITEM_LD_DM and INV_IL_CUR_DM), the recommended steps are described in the following text and shown in the diagram on the following page:

- 1 Populate the INV_IL_CUR_DM table with data. (This population is accomplished the first time that invilddm.ksh runs and an inv position record is processed.)
- 2 Copy this INV_IL_CUR_DM table as the seed to the first partition. (This step is performed automatically by the orapartseed.ksh program. See the chapter, “Compression and partitioning”, for more information about seeding.)
- 3 At this point, only the changed records are added to the INV_ITEM_LD_DM table, whereas the INV_IL_CUR_DM table is a full, uncompressed version that holds the current inventory position as of the last time period.
- 4 When a partition boundary is crossed, the INV_IL_CUR_DM table is copied as the seed to the new partition, via the orapartseed.ksh program.

If you have questions about how to implement partitioning with compression or would like assistance implementing partitioning, contact Retek Customer Support or Retek Services.



Implementing partitioning for compressed inventory tables

How Oracle implements partitions

Oracle 8.0 introduced range partitions. These partitions are split by a range of values on the partition key. Examples include partitions by month, partitions by department number, and partitions by item range. Oracle 8.1 expanded the partitioning options to include hash partitions (spreading the rows across a fixed number of partitions by applying a hash function to the partition key), as well as composite partitioning (a combination of range partitioning and hash partitioning). Retek recommends that clients partition their tables using range partitioning. Retek also recommends that the partition key be the date field in the primary key to allow partitions to be aged out when no longer needed.

Retek's general guideline is that partitioning should be considered for a table that will be 20 GB or larger. Because there is an administrative trade-off between having more partitions to manage and obtaining the benefits of partitioning, partitioning tables smaller than 20 GB should only be considered when specific circumstances dictate. In addition, individual partitions should be kept to 10 GB or less.

The actual physical layout of partitions varies from site to site. A general approach is to put each partition into its own tablespace. This has several advantages:

- Maintenance operations, as well as tablespace recovery, can occur on a partition while other partitions are unaffected
- If manual performance tuning of the data files is being done, tablespaces and their files can be moved around to achieve optimal performance
- If partitions are no longer being updated, their tablespaces can be changed to READ ONLY, which greatly reduces backup requirements

Oracle partitions are ordered from low values to high values. The partition key value for a partition is a non-inclusive upper bound (high value) for that partition. For example, if the SLS_ITEM_LD_DM table is partitioned by month, the high value for the January, 2000, partition is 01-Feb-2000. A low value can always be inserted into the lowest partition. However, a high value may not be able to be inserted depending on the high value of the highest partition. For instance, if the highest partition has a high value of 01-Feb-2000, and the insertion of a record is attempted with a date of 01-Feb-2000, the row cannot be inserted into the table (remember that the high value of 01-Feb-2000, is a *non-inclusive* upper bound). For this reason, Oracle allows a special high value partition with a key of MAXVALUE. Retek recommends that all partitioned tables include a dummy partition with a MAXVALUE high value.

There are special considerations for the partitioning of RDW compressed tables. The following is a brief description of the different partition maintenance commands. Please see the current Oracle documentation set for more details:

- **ADD PARTITION:** adds a new partition to the high end of a partitioned table; because Retek recommends having a MAXVALUE partition, and this is the highest partition, the ADD PARTITION functionality can be achieved by performing a SPLIT of the MAXVALUE partition instead
- **DROP PARTITION:** drops the partition; this is the typical method to delete the oldest partitions (those with the lowest values) as they age to maintain a rolling window of data
- **EXCHANGE PARTITION:** converts a non-partitioned table into a partitioned table or converts a partitioned table into a non-partitioned table
- **MERGE PARTITION:** merges two adjacent partitions into one
- **MOVE PARTITION:** moves a partition to another segment; this is used to defragment a partition or to change its storage characteristics
- **SPLIT PARTITION:** splits an existing partition by adding a new partition at its low end
- **TRUNCATE PARTITION:** removes all rows from the partition

Oracle automatically maintains local index partitions in a 1-to-1 correspondence with their underlying table partitions. Any table partition operations, such as ADD PARTITION, also affect its appropriate index partitions.

Summary

Partitions are useful for breaking up large tables into smaller, more manageable pieces. Retek offers the following recommendations for partitions:

- Consider partitioning tables that will be greater than 20 GB in size or those for which a special need exists that partitioning could benefit
- Keep individual partitions under 10 GB in size
- Use the date as the partition key for range partitioning
- When tables are partitioned, make their indexes local
- Consider putting each partition in its own tablespace
- After updates on a partition cease, consider changing its tablespace to READ ONLY to reduce backup requirements

If partitioning compressed tables, be sure to address their special requirements.

Chapter 5 – RDW program overview

This chapter summarizes RDW's RETL programs. More information about the RETL tool is available in the RETL 10.2 Programmer's Guide.

This chapter references the directory structure set up during RDW product installation. Descriptions of these directories are available in the RDW 10.2 Database Installation Guide.

Program features

RDW 10.2's RETL programs include the following features:

- Program return code
- Program status control files
- Restart and recovery
- Message logging
- Program error file
- Reject data files
- Schema files
- Resource files
- Command line parameters
- Partitioning

Program return code

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

Program status control files

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

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

Naming convention

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

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

- The program name
- The output filename, if one is specified on the command line
- 'status'
- The business virtual date for which the module was run

For example, the program status control file for the invildex program would be named as follows for the batch run of January 5, 2001:

```
$MMHOME/error/invildex.invilddm.txt.status.20010105
```

Restart and recovery

Because RETL processes all records as a set, as opposed to one record at a time, the method for restart and recovery must be different from what was used for Pro*C. The restart and recovery process within RDW 10.2 serves the following two purposes:

- 1 It prevents the loss of data due to program or database failure.
- 2 It increases performance when restarting after a program or database failure by limiting the amount of reprocessing that needs to occur.

DWI restart and recovery

The data warehouse interface (DWI) modules extract from a source transaction database or text file and write to a text file. Modules using a single RETL flow do not require the use of restart and recovery. If the extraction process fails for any reason, the problem can be fixed, and the entire process can be run from the beginning without the loss of data. For a module that takes a text file as its input, the following two choices are available that enable the module to be re-run from the beginning:

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

Note: If you use method 2 for the slsildmex.ksh module, the input file must contain all the records for a given transaction header or the amounts and quantities will be double-counted.

To limit the amount of data that needs to be re-processed, more complicated modules that require the use of multiple RETL flows utilize a bookmark method for restart and recovery. This method allows the module to be restarted at the point of last success and run to completion. The bookmark restart/recovery method incorporates the use of a bookmark flag to indicate which step of the process should be run next. For each step in the process, the bookmark flag is written to and read from a bookmark file.

Note: If the fix for the problem causing the failure requires changing data in the source table or file, then the bookmark file must be removed and the process must be re-run from the beginning in order to extract the changed data.

RDW restart and recovery

Datamart load modules that use a single RETL flow do not require the use of restart and recovery. If the load process fails for any reason, the module can be re-run from the beginning.

More complicated modules that require the use of multiple RETL flows have the potential risk of losing data during a failure. Each individual step is committed once it is successful. Thus, if a later step fails, a rollback is impossible. These modules utilize a bookmark method for restart and recovery. This method allows the module to be restarted at the point of last success and run to completion. The bookmark restart/recovery method incorporates the use of a bookmark flag to indicate which step of the process should be run next. The bookmark flag is written to and read from a bookmark file for each step in the process.

Note: If the fix for the problem causing the failure requires changing data in the source table or file, then the bookmark file must be removed and the process must be re-run from the beginning in order to extract the changed data.

Program control tables

The PROGRAM_CONTROL_DM table is used by RDW 10.2 to determine how to process records for a module. See Chapter 8, “Program reference lists”, for module-specific details about the contents of the PROGRAM_CONTROL_DM table.

The PROGRAM_CONTROL_DM table holds record-keeping information about current program processes. See the RDW 10.2 Data Model for table and column descriptions.

Bookmark file

The name and directory of the restart and recovery bookmark file is set in the configuration files (`dwi_config.env` and `rdw_config.env`). The directory defaults to `$MMHOME/rfx/bookmark`. The naming convention for the bookmark file itself defaults to the following:

- The program name
- The first filename, if one is specified on the command line
- ‘bkm’
- The business virtual date for which the module was run

For example, the bookmark flag for the `invildex` program would be written to the following file for the batch run of January 5, 2001:

```
$MMHOME/rfx/bookmark/invildex.invilddm.txt.bkm.20010105
```

Message logging

Message logs are written daily in a format described in this section.

Daily log file

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

The naming convention of the daily log file defaults to the following:

- The business virtual date for which the modules are run
- ‘.log’

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

```
$MMHOME/log/20010105.log
```

Format

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

```
cusdemogdm 13:20:01: Program Starting...
cusdemogdm 13:20:05: Build update and insert data.
cusdemogdm 13:20:13: Analyze table
rdw10dev.cust_demog_dm_upd
cusdemogdm 13:20:14: Insert/Update target table.
cusdemogdm 13:20:23: Analyze table rdw10dm.cust_demog_dm
cusdemogdm 13:20:27: Program Completed...
```

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

Program error file

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

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

The naming convention for the program’s error file defaults to the following:

- The program name
- The first filename, if one is specified on the command line
- The business virtual date for which the module was run

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

```
$MMHOME/error/slsildm.slsildm.txt.20010105
```

DWI reject files

The phase and season DWI dimension extract modules, as well as most DWI fact extract modules, may produce a reject file if they encounter data-related problems, such as the inability to find data on required lookup tables. The module tries to process all data and then indicates that records were rejected. The idea is that all data problems can be identified in one pass and corrected; then, the module can be re-run to successful completion. If a module does reject records, the reject file is *not* removed, and the client is responsible for removing the reject file before re-running the module.

Note: DWI fact extract modules fail if records are rejected, but the phase and season DWI dimension extract modules do not fail if records are rejected.

The records in the reject file contain an error message and key information from the rejected record. The following example illustrates a record that is rejected due to problems within the currency conversion library:

```
Currency Conversion Failed|101721472|20010309
```

The following example illustrates a record that is rejected due to problems looking up information on a source table:

```
Unable to find item_master record for Item|101721472
```

The name and directory of the reject file is set in the configuration file (`dwi_config.env`). The directory defaults to `$MMHOME/data`.

Note: A directory specific to reject files can be created. The `dwi_config.env` file would need to be changed to point to that directory.

The naming convention for the reject file defaults to the following:

- The program name
- The first filename, if one is specified on the command line
- 'rej'
- The business virtual date for which the module was run

For example, all rejected records for the `slsildmex` program would be placed in the following file for the batch run of January 5, 2001:

```
$MMHOME/data/slsildmex.slsildmdm.txt.rej.20010105
```

Schema files

RETL uses schema files to define an incoming or outgoing dataset. The schema file defines each column's data structure, which is then used within RETL to format/handle the data. For more information about schema files, see the RETL 10.2 Programmer's Guide. For the following reasons, schema file names are hard-coded within each module:

- The schema files should not change on a day-to-day basis.
- The schema used by the DWI extract module must be the same as what is used by the RDW datamart loading module.

Resource files

RDW and DWI Kornshell programs use resource files so that the same RETL programs can run in various language environments. For each language, there is one resource file for the RDW code and one for the DWI code.

Resource files contain hard-coded strings that are used by DWI dimension extract programs and RDW time load programs. The name and directory of the resource file is set in the configuration files (`rdw_config.env` and `dwi_config.env`). The default directory is `$MMHOME/rfx/include`.

The naming convention for the resource file follows the two-letter ISO code standard abbreviation for languages (for example, `en` for English, `fr` for French, `ja` for Japanese, `es` for Spanish, `de` for German, and so on). The RDW code and the DWI code contain different sets of resource files; this fact is reflected in the prefix portion of their applicable naming conventions: `'rdw'` or `'dwi'` (for example, `dwi_en.rsc`, `rdw_fr.rsc`, and so on).

Command line parameters

In order for each RETL module in RDW 10.2 to run, the input/output data file paths and names may need to be passed in at the Unix command line.

DWI

DWI fact extraction modules require passing in the `output_file_path` and `output_file_name` (example 1 below). File-based modules also require an `input_file_path` and `input_file_name` (example 2 below).

Examples 1 and 2 follow:

```
1 ncstuidex output_file_path/output_file_name
2 lptotldex output_file_path/output_file_name
   input_file_path/input_file_name
```

DWI dimension extraction modules do not require passing in any parameters because they have no multi-threading options. The output path/filename defaults to `$DATA_DIR/(DM program name).txt`.

RDW

Datamart base fact load modules require passing in the `input_file_path` and `input_file_name`. Datamart dimension load modules do not require passing in any parameters because the input path/filename defaults to the following:

`$DATA_DIR/(program name).txt`.

See Chapter 8, “Program reference lists”, for a detailed listing of all programs and their command line parameters.

Multi-threading

In contrast to the way in which multi-threading is defined in Unix, RDW 10.2 uses ‘multi-threading’ to refer to the running of a single RETL program multiple times on separate groups of data simultaneously. Multi-threading is only available for DWI fact extraction modules that take a text file as input. Depending upon how it is implemented, multi-threading can reduce the total amount of processing time,

File-based fact extraction modules have to be run once for each input file. A different output file must be specified for each input file. It is the responsibility of the client to set up, as part of the daily batch operation, a process to combine all the resulting text files into one file using the Unix concatenation (‘cat’) command.

For example, if the `lptotldex.ksh` module is run three times for three input files, then it would be called as follows:

```
lptotldex $MMHOME/data/lptotlddm.1000000009
$MMHOME/data/RDWS_1000000009_20020310_20020311

lptotldex $MMHOME/data/lptotlddm.1000000010
$MMHOME/data/RDWS_1000000010_20020310_20020311

lptotldex $MMHOME/data/lptotlddm.1000000011
$MMHOME/data/RDWS_1000000011_20020310_20020311
```

To concatenate the three output files, run the following command in the `$MMHOME/data` directory:

```
cat lptotlddm.1000000009 lptotlddm.1000000010
lptotlddm.1000000011 > lptotlddm.txt
```

In this example, `lptotlddm.txt` becomes the combined text file. This text file is then loaded by an RDW RETL batch program, which expects all of the facts for today's loss-prevention store totals datamart to be included in one text file. For more information, see the command line parameters column on the chart in Chapter 8.

Partitioning

RETL partitioning divides the data into multiple segments, or partitions, based upon the number of logical partitions defined in RETL. The default recommendation is to set the number of logical partitions equal to the number of available CPU nodes. Each processor is responsible for a portion of a dataset, rather than the entire dataset. As a result of this partition load method, the processing of the entire dataset is much faster than in a single-processor environment.

Partitioning operators work closely with parallel operators, so that the detailed operations of partitioning and parallelism are hidden from the application user. See the RETL 10.2 Programmer's Guide for details on setting the number of partitions and determining which operators default to serial and which to parallel.

Typical run and debugging situations

The following examples illustrate typical run and debugging situations for each type of program within RDW 10.2. The log, error, and so on file names referenced below assume that the module is run on the business virtual date of March 9, 2001. See the previously described naming conventions for the location of each file.

DWI dimension extract

To run orglocex.ksh:

- 1 Change directories to \$MMHOME/rfx/src.
- 2 At a Unix prompt enter:

```
%orglocex.ksh
```

If the module runs successfully, the following results:

- 1 **Log file:** Today's log file, 20010309.log, contains the messages "Program started ..." and "Program completed successfully" for orglocex.
- 2 **Data:** The orglocdm.txt file exists in the data directory and contains the extracted records.
- 3 **Error file:** The program's error file, orglocex.20010309, contains the standard RETL flow (ending with "All threads complete" and "Flow ran successfully") and no additional error messages.
- 4 **Program status control:** The program status control file, orglocex.status.20010309, does not exist.
- 5 **Reject file:** The reject file, orglocex.rej.20010309, does not exist.

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

- 1 **Log file:** Today's log file, 20010309.log, does not contain the "Program completed successfully" message for orglocex.
- 2 **Data:** The orglocdm.txt file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, orglocex.20010309, may contain an error message.
- 4 **Program status control:** The program status control file, orglocex.status.20010309, exists.
- 5 **Reject file:** The reject file, orglocex.status.20010309, does not exist because this module does not reject records.
- 6 **Bookmark file:** The bookmark file, orglocex.bkm.20010309, does not exist because this module does not utilize restart and recovery.

To re-run the module, perform the following actions:

- 1 Determine and fix the problem causing the error.
- 2 Remove the program's status control file.
- 3 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%orglocex.ksh
```


DWI fact extract

Table-based

Table-based DWI fact extract modules read their source data from a database table and write their output to a flat file.

To run `invildex.ksh`, change directories to `$MMHOME/rfx/src`. At the Unix prompt, enter:

```
%invildex.ksh $MMHOME/data/invilddm.txt
```

If the module runs successfully, the following results:

- 1 **Log file:** Today's log file, `20010309.log`, contains "Program started ...", various informational messages, "Number of records in <output path and filename> = <number of records processed>", and "Program completed successfully" messages for `invildex`.
- 2 **Data:** The `invilddm.txt` file exists in the data directory and contains the extracted records.
- 3 **Error file:** The program's error file, `invildex.invilddm.txt.20010309`, contains the program's standard RETL flow (with "All threads complete" and "Flow ran successfully"), standard database output when dropping tables, and no error messages.
- 4 **Program status control:** The program status control file, `invildex.invilddm.txt.status.20010309`, does not exist.
- 5 **Reject file:** The reject file, `invildex.invilddm.txt.rej.20010309`, does not exist.
- 6 **Bookmark file:** The bookmark file, `invildex.invilddm.txt.bkm.20010309`, does not exist.

If the module does *not* run successfully because records were rejected, the following results:

- 1 **Log file:** Today's log file, `20010309.log`, does not contain the "Program completed successfully" message for `invildex`. The log file contains the messages "Number of records in <reject path and filename> = <number of rejected records>" and "ERROR: **Records rejected by invildex".
- 2 **Data:** The `invilddm.txt` file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, `invildex.invilddm.txt.20010309`, contains the program's standard RETL flow and possibly additional error messages.
- 4 **Program status control:** The program status control file, `invildex.invilddm.txt.status.20010309`, exists.
- 5 **Reject file:** A reject file, `invildex.invilddm.txt.rej.20010309`, exists. Each rejected record contains a detailed error message.
- 6 **Bookmark file:** The bookmark file, `invildex.invilddm.txt.bkm.20010309`, does not exist.

If the module does *not* run successfully for a reason other than rejected records, the following results:

- 1 **Log file:** Today's log file, 20010309.log, does not contain the "Program completed successfully" message but may contain other informational messages for invildex.
- 2 **Data:** The invilddm.txt file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, invildex.invilddm.txt.20010309, contains the program's RETL flow, and any additional error messages.
- 4 **Program status control:** The program status control file, invildex.invilddm.txt.status.20010309, exists.
- 5 **Reject file:** The reject file, invildex.invilddm.txt.rej.20010309, may or may not exist for invildex.
- 6 **Bookmark file:** The bookmark file, invildex.invilddm.txt.bkm.20010309, exists. No bookmark file is created if the module did not make it past the first unit of work within the module or if the module does not use restart and recovery.

To re-run the module, perform the following actions:

- 1 Determine and fix the problem causing the error.
- 2 If you wish to re-run the module from the beginning, remove the program's bookmark file.
- 3 Remove the program's control status file.
- 4 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%invildex.ksh $MMHOME/data/invilddm.txt
```

File-based

File-based DWI fact extract modules read their source data from a flat file and write their output to a flat file.

To run lptotldex.ksh, change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%lptotldex.ksh $MMHOME/data/lptotlddm.txt
$MMHOME/data/RDWS_1000000009_20020310_20020311
```

If the module runs successfully, the following results:

- 1 **Log file:** Today's log file, 20010309.log, contains "Program started ...", various informational messages, "Number of records in <output path and filename> = <number of records processed>", and "Program completed successfully" messages for lptotldex.
- 2 **Data:** The lptotlddm.txt file exists in the data directory and contains the extracted records.

- 3 **Error file:** The program's error file, lptotldex.lptotlddm.txt.20010309, contains the program's standard RETL flow (with "All threads complete" and "Flow ran successfully"), standard database output when dropping tables, and no additional error messages.
- 4 **Program status control:** The program status control file, lptotldex.lptotlddm.txt.status.20010309, exists.
- 5 **Reject file:** The reject file, lptotldex.lptotlddm.txt.rej.20010309, does not exist.
- 6 **Bookmark file:** The bookmark file, lptotldex.lptotlddm.txt.bkm.20010309, does not exist.

If the module does *not* run successfully because records were rejected, the following results:

- 1 **Log file:** Today's log file, 20010309.log, does not have the "Program completed successfully" message for lptotldex. The log file contains the messages "Number of records in <reject path and filename> = <number of rejected records>" and "ERROR: **Records rejected by lptotldex".
- 2 **Data:** The lptotlddm.txt file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, lptotldex.lptotlddm.txt.20010309, contains the program's standard RETL flow (with "All threads complete" and "Flow ran successfully"), standard database output when dropping tables, and no additional error messages.
- 4 **Program status control:** The program status control file, lptotldex.lptotlddm.txt.status.20010309, exists.
- 5 **Reject file:** A reject file, lptotldex.lptotlddm.txt.rej.20010309, exists. Each rejected record contains a detailed error message.
- 6 **Bookmark file:** The bookmark file, lptotldex.lptotlddm.txt.bkm.20010309, does not exist.

If the module does *not* run successfully for a reason other than rejected records, the following results:

- 1 **Log file:** Today's log file, 20010309.log, has a "...Program failed..." message for lptotldex.
- 2 **Data:** The lptotlddm.txt file may exist in the data directory but may not contain all the extracted records.
- 3 **Error file:** The program's error file, lptotldex.lptotlddm.txt.20010309, contains the program's RETL flow, and any additional error messages.
- 4 **Program status control:** The program status control file, lptotldex.lptotlddm.txt.status.20010309, exists.
- 5 **Reject file:** A reject file, lptotldex.lptotlddm.txt.rej.20010309, may exist.
- 6 **Bookmark file:** The bookmark file, lptotldex.lptotlddm.txt.bkm.20010309, does not exist. (If the module uses restart and recovery, a bookmark file may exist.)

To re-run the module, perform the following actions:

- 1 Determine and fix the problem causing the error.
- 2 If you wish to re-run the module from the beginning, remove the program's bookmark file.
- 3 Remove the program's control status file.
- 4 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%lptotldex.ksh $MMHOME/data/lptotlddm.txt
$MMHOME/data/RDWS_1000000009_20020310_20020311
```

RDW dimension load

To run prdpimdm.ksh:

- 1 Change directories to \$MMHOME/rfx/src.
- 2 At a Unix prompt, enter:

```
%prdpimdm.ksh
```

If the module runs successfully, the following results:

- 1 **Log file:** Today's log file, 20010309.log, contains "Program started ...", various informational messages, and "Program Completed ..." messages for prdpimdm.
- 2 **Data:** The records from the source file \$MMHOME/data/prdpimdm.txt are loaded into the target table.
- 3 **Error file:** The program's error file, prdpimdm.20010309, contains the program's standard RETL flow (with "All threads complete" and "Flow ran successfully"), standard database output for dropping/updating tables, and no additional error messages.
- 4 **Program status control:** The PROGRAM_STATUS_DM table is updated to 'completed' where program_name = prdpimdm and file_name = \$MMHOME/data/prdpimdm.txt.
- 5 **Reject file:** Reject files are not created for RDW modules.
- 6 **Bookmark file:** The bookmark file, prdpimdm.bkm.20010309, does not exist.

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

- 1 **Log file:** Today's log file, 20010309.log, does not have the "Program Completed ..." message for prdpimdm.
- 2 **Data:** Some of the records from the source file \$MMHOME/data/prdpimdm.txt may be loaded into the target table.
- 3 **Error file:** The program's error file, prdpimdm.20010309, contains the program's RETL flow and any additional error messages.
- 4 **Program status control:** The PROGRAM_STATUS_DM table is updated to 'error' where program_name = prdpimdm and file_name = \$MMHOME/data/prdpimdm.txt.
- 5 **Reject file:** Reject files are not created for RDW modules.
- 6 **Bookmark file:** The bookmark file, prdpimdm.bkm.20010309, may exist. No bookmark file is created if the module did not make it past the first unit of work within the module or the module does not use restart and recovery.

To re-run the module, perform the following actions:

- 1 Determine and fix the problem causing the error.
- 2 If you wish to re-run the module from the beginning, remove the program's bookmark file.
- 3 Update the PROGRAM_STATUS_DM table to 'ready' where program_name = prdpimdm and file_name = \$MMHOME/data/prdpimdm.txt.
- 4 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%prdpimdm.ksh
```

RDW fact load

To run vchreschddm.ksh:

- 1 Change directories to \$MMHOME/rfx/src.
- 2 At a Unix prompt, enter:

```
%vchreschddm.ksh $MMHOME/data/vchreschddm.txt
```

If the module runs successfully, the following results:

- 1 **Log file:** Today's log file, 20010309.log, contains "Program started ...", various informational messages, and "Program completed successfully" messages for vchreschddm.
- 2 **Data:** The records from the source file \$MMHOME/data/vchreschddm.txt are loaded into the target table.
- 3 **Error file:** The program's error file, vchreschddm.vchreschddm.txt.20010309, contains the program's standard RETL flow (with "All threads complete" and "Flow ran successfully"), standard database output for updating tables, and no additional error messages.

- 4 **Program status control:** The PROGRAM_STATUS_DM table is updated to 'completed' where program_name = vchreschddm and file_name = \$MMHOME/data/vchreschddm.txt.
- 5 **Reject file:** Reject files are not created for RDW modules.
- 6 **Bookmark file:** The bookmark file, vchreschddm.vchreschddm.txt.bkm.20010309, may exist. No bookmark file is created if the module did not make it past the first unit of work within the module or if the module does not use restart and recovery.

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

- 1 **Log file:** Today's log file does not contain the "Program completed successfully" message for vchreschddm.
- 2 **Data:** Some of the records from the source file \$MMHOME/data/vchreschddm.txt may be loaded into the target table.
- 3 **Error file:** The program's error file, vchreschddm.vchreschddm.txt.20010309, contains the program's RETL flow, and any additional error messages.
- 4 **Program status control:** The PROGRAM_STATUS_DM table is updated to 'error' where program_name = vchreschddm and file_name = \$MMHOME/data/vchreschddm.txt.
- 5 **Reject file:** Reject files are not created for RDW modules.
- 6 **Bookmark file:** The bookmark file, vchreschddm.vchreschddm.txt.bkm.20010309, may exist. No bookmark file is created if the module did not make it past the first unit of work within the module or if the module does not use restart and recovery.

To re-run the module:

- 1 Determine and fix the problem causing the error.
- 2 If you wish to re-run the module from the beginning, remove the program's bookmark file.
- 3 Update the PROGRAM_STATUS_DM table to 'ready' where program_name = vchreschddm and file_name = \$MMHOME/data/vchreschddm.txt.
- 4 Change directories to \$MMHOME/rfx/src. At a Unix prompt, enter:

```
%vchreschddm.ksh $MMHOME/data/vchreschddm.txt
```

Chapter 6 – RDW interfaces

This chapter provides a functional summary of data interfaces with RDW, which can receive data through these interfaces in one of two ways.

First, RDW can extract data from a source system through the use of its data warehouse interface (DWI) code (the sources are RMS and ReSA). The APIs/schema files associated with this method are illustrated in the “Extraction and load” subsections for both dimensions and facts in Appendix A, “Application programming interface (API) flat file specifications.” The data from the DWI extraction is then loaded to the RDW side using the same schema files. More information about schema files can be found in the RETL 10.2 Programmer’s Guide.

A second way to load data to RDW is through externally populated interfaces provided by the client or other Retek applications (such as TopPlan). RDW code does not extract this data, but does load this external data to RDW. The APIs/schema files associated with this method are illustrated in the “Load only” subsections for both dimensions and facts in Appendix A, “Application programming interface (API) flat file specifications.”

The following interfaces are described in this section:

- Extract and load
 - Retek Merchandising System (RMS)
 - Dimension data
 - Fact data
 - Retek Sales Audit (ReSA)
- Load only
 - Retek TopPlan
 - Retek Customer Order Management (RCOM)
 - Client-supplied data:
 - Customer account dimension
 - Customer geographic dimension
 - Customer and product cluster dimensions
 - Plan Season dimension
 - Market data facts and dimensions
 - Space allocation facts
 - Store traffic facts
 - Two of six supplier compliance facts text files
 - Quality control
 - Missed scheduled deliveries
 - Loaded at install: Voucher age dimension and Time like for like transformations

All data comes into RDW as text files. See Appendix A, “Application programming interface (API) text file specifications”, for a complete list of RDW’s API specifications and their business requirements.

Retek Merchandising System

The Retek Merchandising System (RMS) can be the principle source of RDW's dimension and fact data. RMS is the retail client's central transactional processing system. RMS data feeds to RDW can be broken down into dimensions and facts. General descriptions of each are contained in this section.

Dimension data

RMS can be the sole source of organization and product dimension data, and it supplies the majority of other dimension data as well. If the client does not use RMS, dimension data is loaded directly from another source.

RDW's dimension data process extracts current dimension data from the RMS using RETL scripts, as part of the data warehouse interface (DWI). The extracted data is outputted to text files. After these text files are moved to the RDW server, RETL then compares the data in the text file with the historical dimension data in RDW, and thereafter inserts/updates the dimension changes back into RDW. This comparison eliminates the need to capture dimension changes as they occur in the source system during the day.

RMS supplied dimensions include the following: Company, Competitor, Currency Code, Employee, Item-Location Trait Cross, Item-Supplier-Location Cross, Organization, Product (including attributes, such as differentiators), Product Season, Promotion, Reason, Regionality, Sub-Transaction Type, Supplier, Tender Type, and Total Type.

RMS can be the source of one of the two types of time that RDW supports: fiscal 454 time. Clients can supply the other RDW-supported time functionality: 13 period time. For more information on how time is loaded to RDW, see the RDW 10.2 Database Installation Guide.

Fact data

If the client does not use RMS, data is loaded directly from another source.

RDW's fact data process extracts fact data from RMS using RETL scripts that run within the RMS batch-processing schedule as part of the DWI. The extracted data is outputted to text files. After these text files are moved to the RDW server, RETL takes the data in the text files and performs the appropriate transformation, inserts and updates to the fact datamart tables.

RMS supplied facts include the following: Competitor Pricing, Cost, Exchange Rates, Inventory Adjustments, Inventory Position, Inventory Receipts, Inventory Transfers, Markdowns, Net Cost, Pricing, Profit on Base Cost, Return to Vendor, Sales Forecasts, Stock Ledger, Supplier Availability, Supplier Compliance, Supplier Contract, Supplier Invoice Cost, Unavailable Inventory, and Currency Exchange Rates.

A note about local currency and facts

Many RDW clients conduct business in a multi-currency environment. While querying sales facts, for instance, a client may want to see the values in the common local currency of a group of stores in one country, or see values aggregated across all their stores from a number of countries. In order to provide clients both accuracy and flexibility in storing currency values, both local and primary currency values are stored in most of the RDW 10.2 fact tables. A client using multiple currencies would have any fact that is stored by `loc_key` held in that location's local currency for that day, side-by-side with a column for that fact converted to primary currency. A client using only one currency would have only the primary currency column populated, leaving the local column null. This currency storage strategy is accomplished by either RETL fact extraction code or legacy fact interfaces, which generate text files that include both local and primary currency values for loading into the datamart tables.

Retek Sales Audit

Retek Sales Audit (ReSA) is a flow through application that accepts 'raw' point of sale information and provides 'clean' data to downstream applications, such as the RDW. If a client does not use ReSA, data is loaded directly from another source.

Retek Sales Audit writes four flat (ASCII text) files for RDW—one each for transaction item data (file type RDWT), transaction tender data (RDWF), store total data (RDWS), and cashier or register over or short data (RDWC). Each of these files is then made available for processing by RETL scripts to extract data. A part of the fact processing, these RETL scripts run within the RMS batch-processing schedule. The extracted data is outputted to text files. On the RDW servers, RETL takes the data in the text files and performs the appropriate inserts and updates to the fact datamart tables.

ReSA supplied facts include the following: Sales and Return Transactions (including Pack Sales), Sales Productivity, Loss Prevention Transactions, and Loss Prevention Totals (Tender Transactions, Cashier Over or Short, User-Defined Totals).

Register dimension data is derived from ReSA RDWF (tender transaction) file, by way of the `ttldmdm.ksh` fact DM script.

In addition to the four flat files described previously, ReSA serves as a source for voucher fact data for RDW. Three DWI programs extract fact data for voucher movement, escheated vouchers, and outstanding vouchers. See Chapter 8 for more details about these programs.

Retek TopPlan

Retek TopPlan provides planning data such as planned sales to a retailer. If the client does not use Retek TopPlan, planning data is loaded to RDW directly from another source.

TopPlan can serve as the source of planning fact data. TopPlan writes two text files for RDW: ploblwdm.txt for original planning, and plcblwdm.txt for current planning. After these text files are moved to the RDW server, both of these files are made available for RETL to take the data in the text files and perform the appropriate inserts and updates to the planning fact datamart tables. Data from TopPlan does not need to be loaded daily; it can be loaded periodically.

Retek Customer Order Management

Retek Customer Order Management (RCOM) is a centralized solution across all channels, that manages customer interactions, purchases, history on the Web, call center/catalogue, kiosk, and the stores using one common infrastructure with a single view of inventory. If the client does not use RCOM, data is loaded directly from another source.

RCOM can serve as the source of customer and customer demographic dimensional data. RCOM writes one text file for RDW: custdm.txt. After this text file is moved to the RDW server, RETL then compares the data in the text file with the historical data in RDW, and thereafter inserts the entire dimension datamart table into RDW.

Client-supplied data

The RDW provides programs and tables for the areas of functionality described in this section. However, there currently is no Retek source system available to provide data for these functional areas. Clients need to supply the data via text files. These text files will be used as the inputs to process and load data into RDW datamart tables. To see the location of client-supplied data in the RDW program schedule, see Chapter 7. For more business content information regarding the following functional areas of client-supplied data, see Appendix A:

- Customer account dimension
- Customer geographic dimension
- Customer and product cluster dimension
- Plan season dimension
- Market data facts and dimensions
- Space allocation facts
- Store traffic facts
- Two of six supplier compliance facts text files
 - Quality control
 - Missed schedule deliveries

The tables representing the following areas of functionality are loaded once at installation: voucher age dimension and time like for like transformations. See the RDW 10.2 Database Installation Guide for more information.

Chapter 7 – Program flow diagrams

This chapter presents flow diagrams for all RDW 10.2 dimension and fact data processing from source systems. Included are descriptions of the source system's program or output file, as the case may be, that is required to run or be present, along with the RDW program or process that interfaces with the source. After initial interface processing of the source, the diagrams illustrate the flow of the data into the respective datamarts.

Before setting up an RDW program schedule, familiarize yourself with the functional and technical constraints associated with each program. Read through the RDW 10.2 Database Installation Guide and Chapter 8, "Program reference lists", of this operations guide for more details.

Batch scheduling

The following explains the order constraints of the RDW batch schedule. This section includes:

- Overall batch schedule, including schedule timing and when to run programs—daily, weekly, ad hoc, and so on
- Functional interdependencies, including functional constraints, such as that fact modules must run after dimension modules

Setting up the batch schedule

Note: The number of modules that can be run in parallel at any given time is dependent upon the client's hardware capacity.

The following discussion applies to Oracle and Teradata RDW clients. DB2 clients must refer to the "Batch schedule for DB2 clients" section later in this chapter. On a typical batch production run, the pre-batch maintenance modules must always run first. What runs next, as long as the client follows the batch dependencies in the flow diagrams, is up to the client.

For example:

- On the DWI side, all dimension and fact extract modules can be run in parallel. (Note, however, that some DWI modules have RMS pre-dependencies, and some RMS modules are dependent upon DWI modules.) Extract modules from the product dimension (prdcmpex.ksh, prditmex.ksh, and so on) can be run in parallel with fact extract modules (slsildmex.ksh, prcildex.ksh, and so on). Because of the extract modules' ability to be run in parallel, it does not matter which module goes first (assuming no interdependencies are noted on the batch flows).
- On the RDW side, product dimension modules, such as prditmdm.ksh and prditlmdm.ksh, can be run in parallel after their respective pre-dependencies. Fact modules, such as prcilddm.ksh, can run in parallel with other, unrelated fact modules provided their respective pre-dependencies (including dimension predecessors) complete successfully first.

The batch flows on the following pages are best read from top to bottom. Such a review of the RDW batch schedule allows clients to both set up module dependencies and to optimize their batch window through the concurrent running of unrelated modules.

dwi_config.env settings

On the DWI side, make sure to review the environmental parameters in the dwi_config.env file before executing batch modules. See the RDW 10.2 Database Installation Guide for more about this topic.

rdw_config.env settings

The RDW 10.2 Database Installation Guide refer to two important RETL environment variables that the client must set in the rdw_config.env file: LOAD_TYPE and SCHEDULE_TYPE.

Note: Clients must weigh the performance benefit of these settings before running their batch schedule.

- LOAD_TYPE refers to the load method that RETL uses to load data to the database, and is only used with Oracle or DB2 DBMS.
 - **LOAD_TYPE=conventional:** loads the data using the conventional SQL-loader method for Oracle, or the DB2LOADER utility for DB2.
 - **LOAD_TYPE=direct:** loads the data using the direct SQL_loader method for Oracle, or the Autoloader utility for DB2. Note that there is one exception to this rule: For DB2 clients, even when LOAD_TYPE is set to direct, all dimension modules (except dimension matrix modules) continue to use the DB2LOADER utility, not the Autoloader.
- The SCHEDULE_TYPE is only used for DB2 clients, and only affects DBMS loading where LOAD_TYPE=direct. If LOAD_TYPE=conventional, SCHEDULE_TYPE is ignored. Valid values for SCHEDULE_TYPE are sequential or parallel.
 - When SCHEDULE_TYPE is set to sequential, the following assumptions apply:
 - There is one tablespace for all dimension tables
 - There is one tablespace for all dimension matrix tables
 - There is one tablespace for each fact table
 - There are three user data tablespaces for temp tables

DB2 tablespaces are set up in this manner per the RDW 10.2 base install. Even though all dimension modules (except dimension matrix) can be scheduled to run parallel, all dimension matrix modules and all fact modules must be run one at a time.

- When SCHEDULE_TYPE is set to parallel, the dimension matrix modules and fact modules can be running in parallel, but a tablespace must be created for each dimension matrix table and each fact temp table. This step requires the slight customization of RDW 10.2 install scripts/procedures, and some potential customization of the RDW 10.2 RETL code. Contact Retek Customer Care for assistance with this type of custom work.

RMS, ReSA and the RDW batch schedule

The RDW's data warehouse interface (DWI) extraction modules run in the RMS batch cycle, and are dependent on some RMS and ReSA modules to provide data for processing (see the descriptions of the individual modules for details). Some RMS modules are dependent on DWI modules. Most DWI extraction programs run after Phase 2 of the RMS batch cycle is completed. All DWI modules must run before the RMS vdate is incremented to the next day; otherwise, today's facts will not get extracted from RMS.

Note: RDW assumes that all RMS fact processing (via batch, forms and/or the RIB) for today is complete in RMS before the DWI fact extracts run. If RMS fact processing is *not* complete, those applicable facts wait until tomorrow to be processed by the DWI code.

Within RDW 10.2, programs are scheduled on a dependency basis rather than in phases, as they are in RMS. These dependencies are described in the program flow diagrams.

TopPlan to RDW scheduling

Original and current plan data from Retek TopPlan are only loaded into RDW periodically. See Chapter 6, "RDW Interfaces", for more details on the flow of data from TopPlan to RDW.

Data from undefined sources

There are no pre-defined sources for some functional areas such as Geographic Dimension, Space Allocation, and Store Traffic fact data. User-defined processes must populate the text files for these areas before their respective loading programs run.

RDW batch schedule for DB2 clients only

Because of DB2's unique loading requirements, RDW uses both the db2write and autoload utilities. The db2write utilities are used to write smaller sets of data. To help enhance load performance speed, the autoload utilities are used for larger sets of data.

Note that the use of autoloader has important ramifications with regard to the client's ability to read or write in parallel. When autoloader is used, the utilities lock the entire table space. Because any tables that reside in the locked tablespace become inaccessible, sequential processing becomes mandatory.

Note that in the base setup of RDW 10.2, modules are set up and scheduled to run in the following ways:

- Dimension modules use db2write utilities and can be run in parallel.
- Dimension matrix modules use autoload utilities and must run in sequential order.
- As shipped in base, all fact modules use autoload utilities and must run in sequential order (see the section, "config.env settings" earlier in this chapter). Even though the fact modules are running in sequential order, some modules use multiple temp tables for reading/writing. Those temp tables need to sit in separate tablespaces.

Note: If a client wants to run different fact datamarts in parallel, it must configure its user tablespaces to its specific processing needs, as well as modify the base code to write to the correct user tablespace.

For more information about the db2write and the autoload utilities, refer to DB2 documentation.

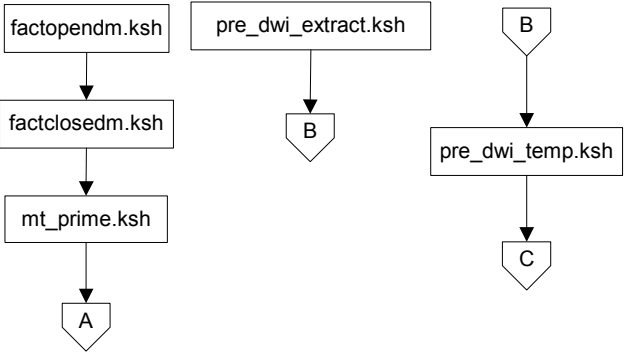
Program flow diagrams

Diagrams of RDW 10.2's program flows begin on the next page.

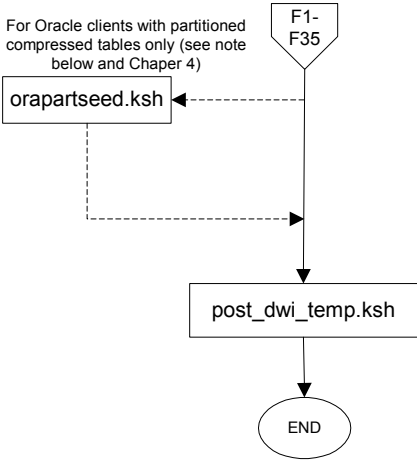
Legend: RDW 10.2 dimension programs



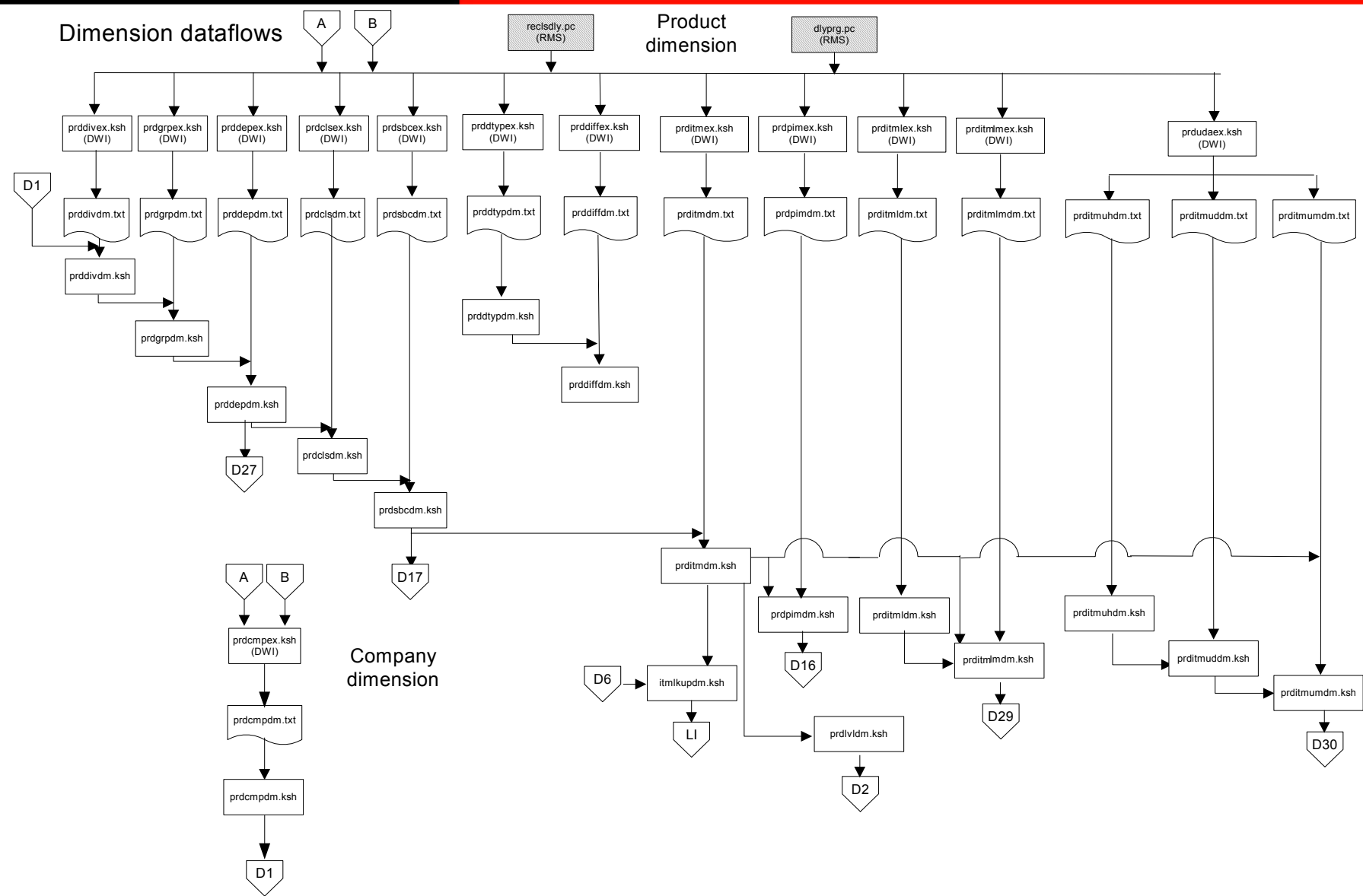
Pre-batch
maintenance



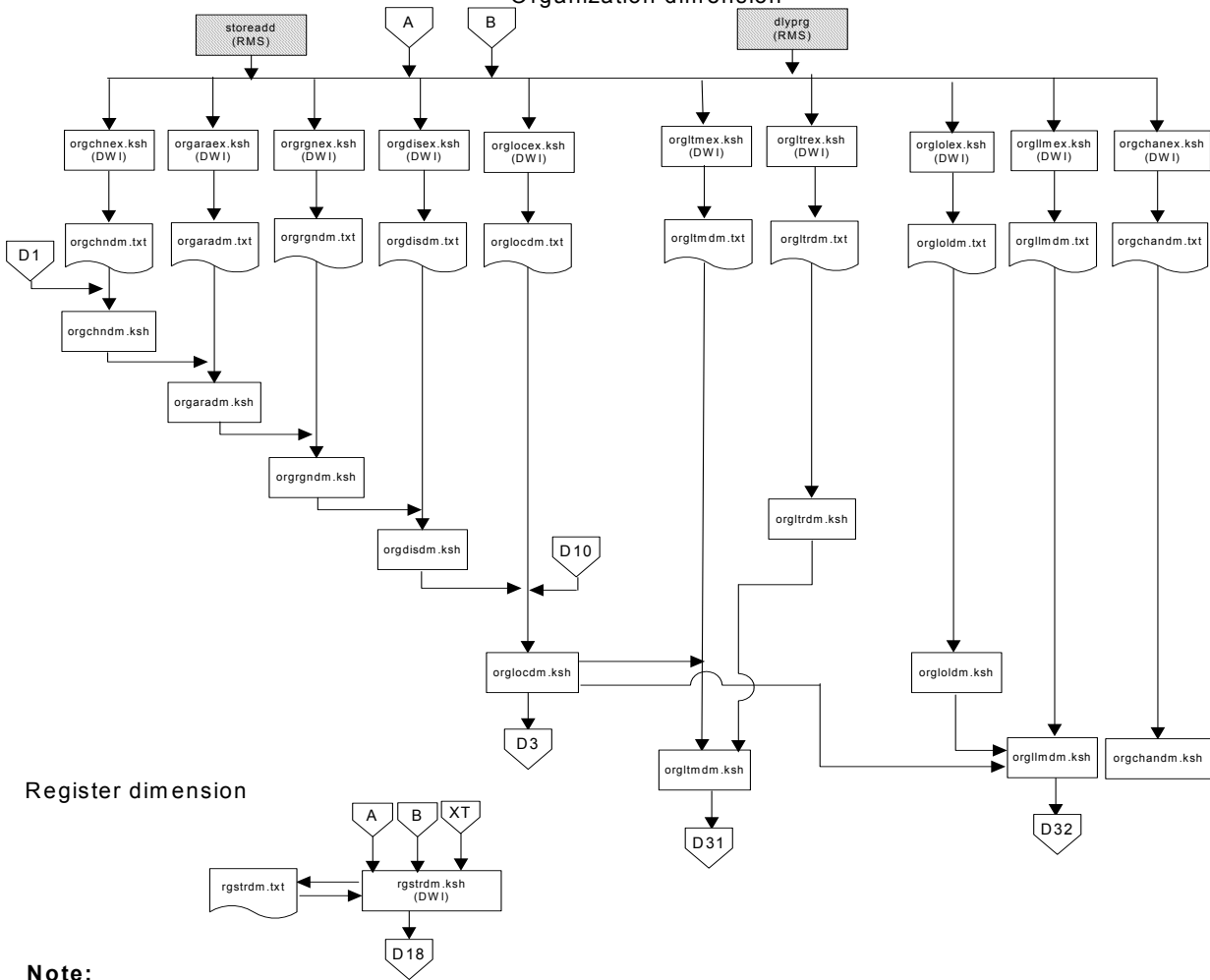
Post-batch
maintenance



Note:
Orapartseed.ksh is an optional program used by Oracle clients only. The program affects compressed, partitioned datamart tables. See the chapter, "Compression and partitioning," for a more detailed explanation of seeding.



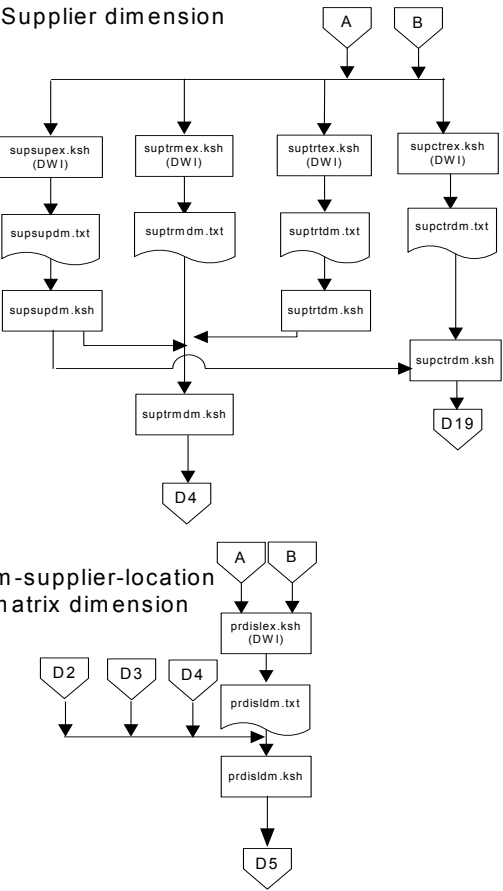
Dimension dataflows



Register dimension

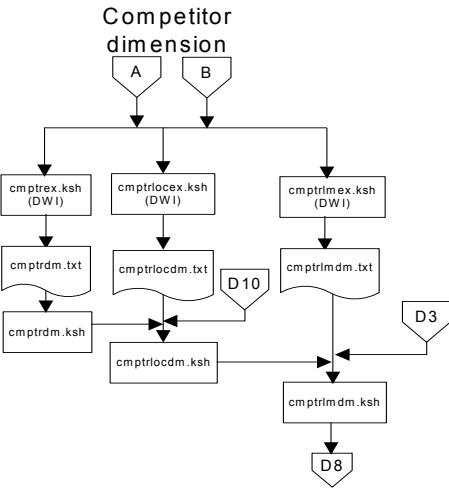
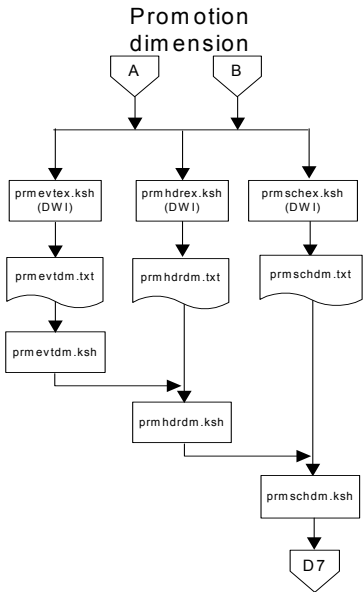
Note:
rgstrdm.txt internally
generated by rgstrdm.ksh

Supplier dimension

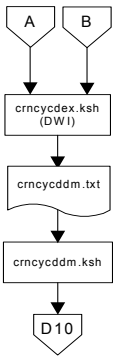


Item-supplier-location
matrix dimension

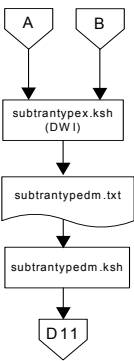
Dimension dataflows



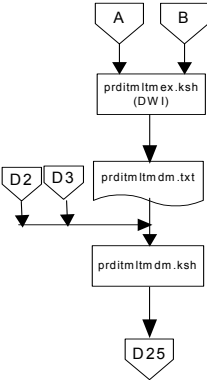
Currency code dimension



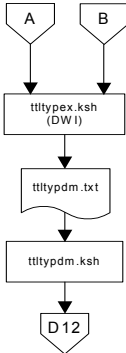
Sub-transaction type dimension



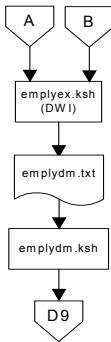
Item-location trait matrix dimension



ReSA total type dimension



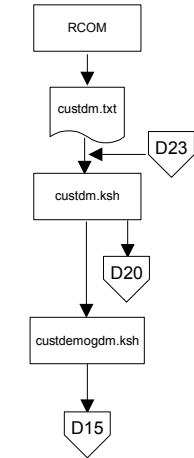
Employee dimension



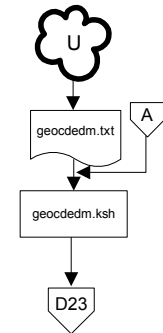
Dimension dataflows

Customer and customer demographic dimension

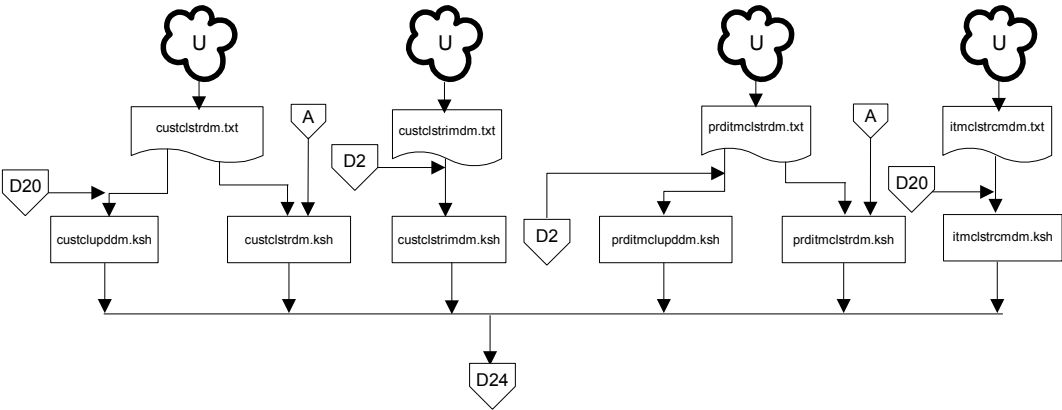
Note:
Text file originates in Retek Customer Order Management (RCOM)



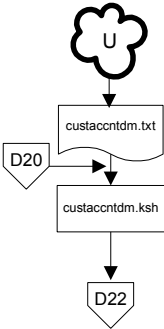
Customer geographic dimension



Customer and product clustering dimension

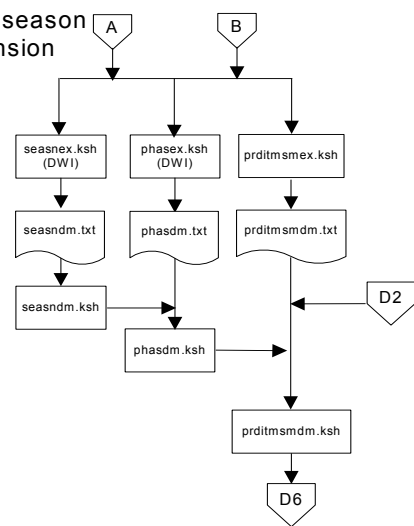


Customer account dimension

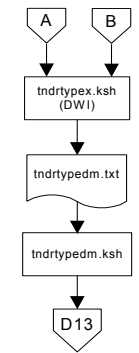


Dimension dataflows

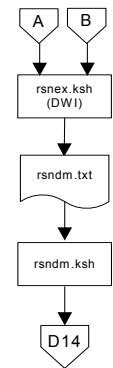
Product season dimension



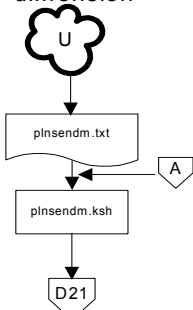
Tender type dimension



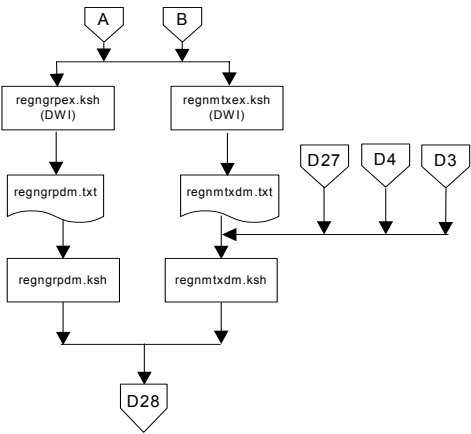
Reason dimension



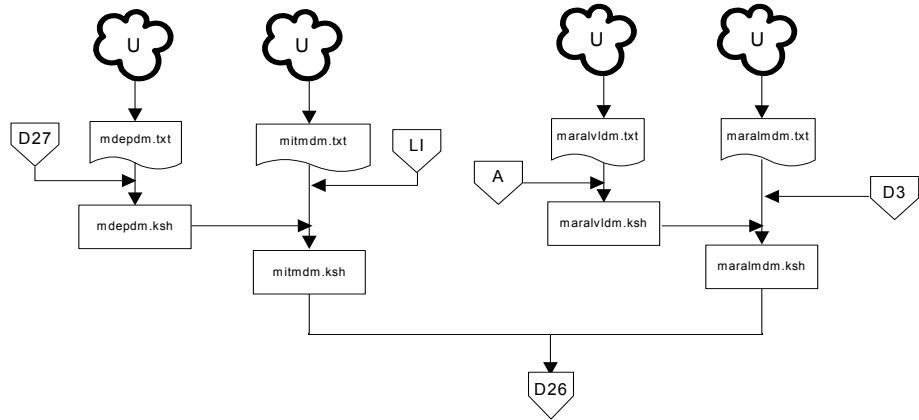
Plan season dimension



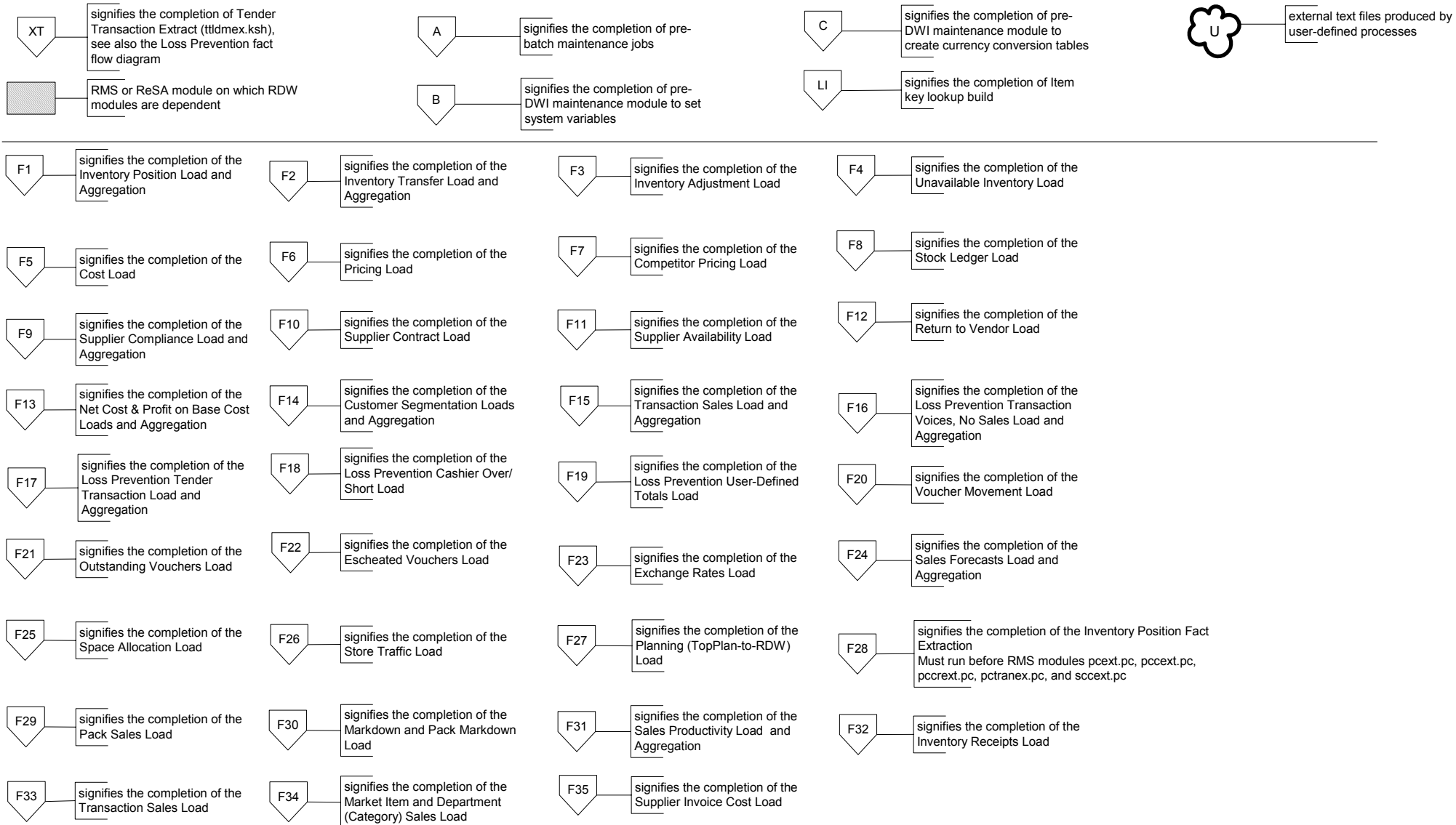
Regionality dimension



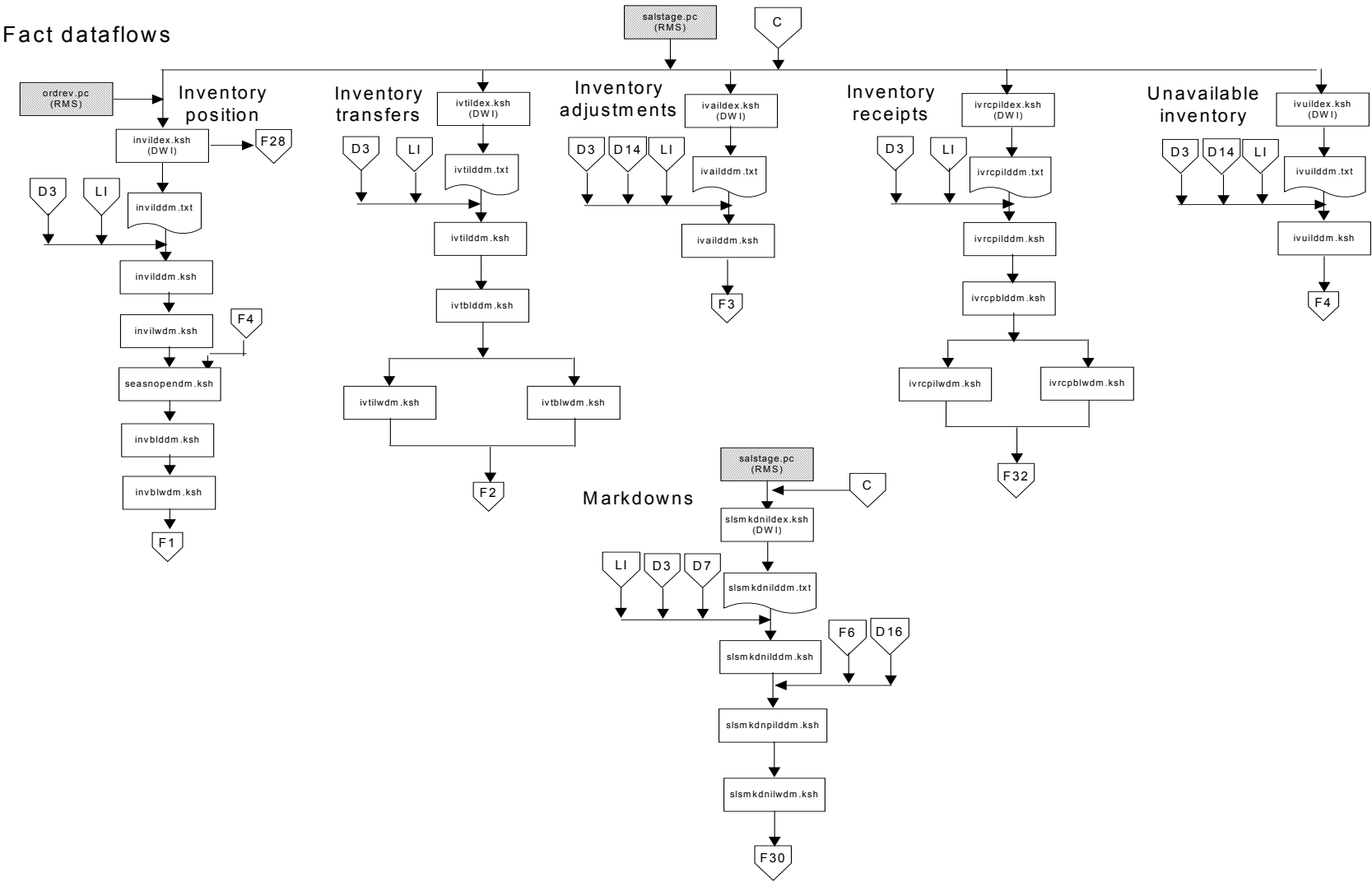
Market data dimension



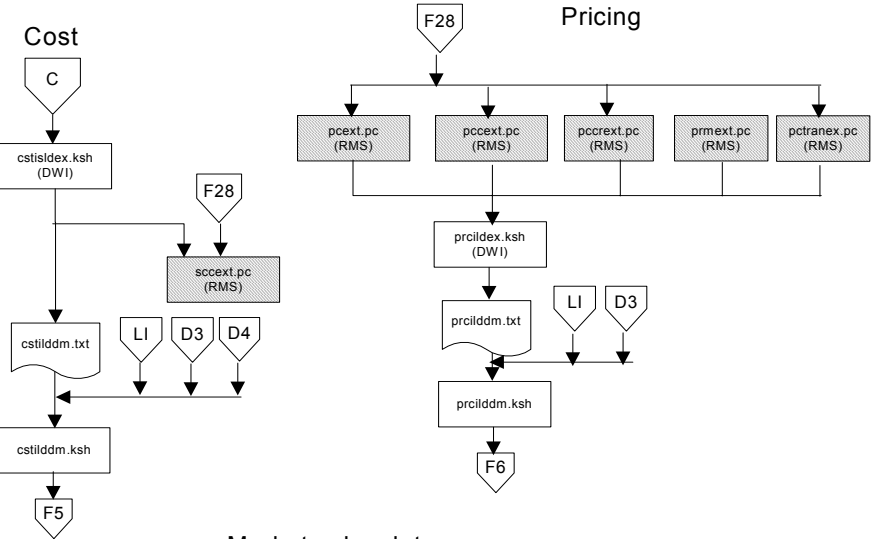
Legend: RDW 10.2 fact programs



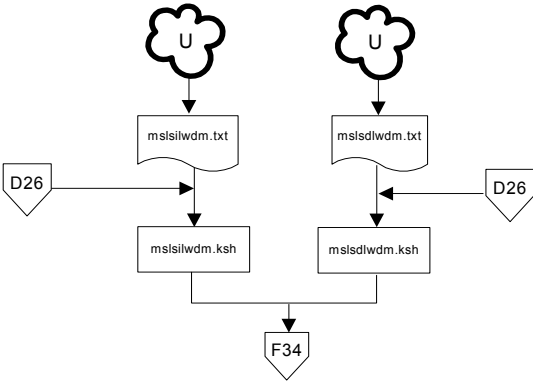
Fact dataflows



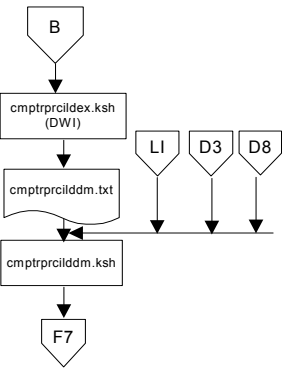
Fact dataflows



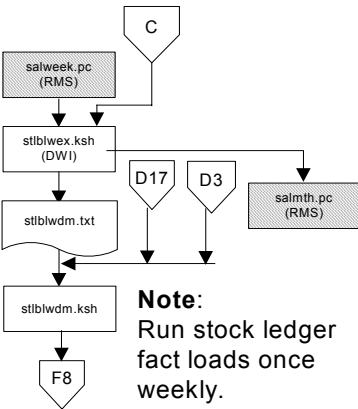
Market sales data



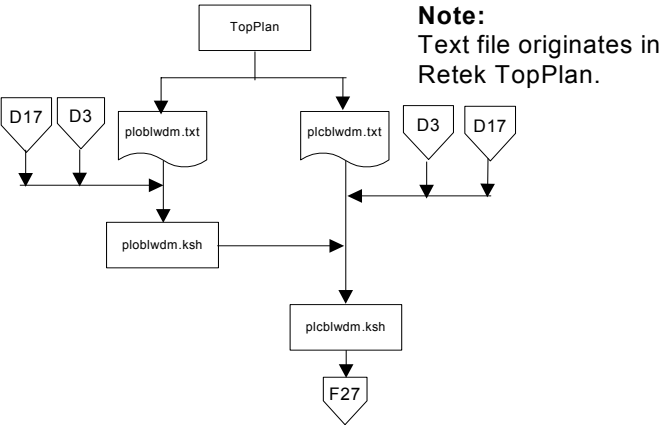
Competitor pricing



Stock ledger

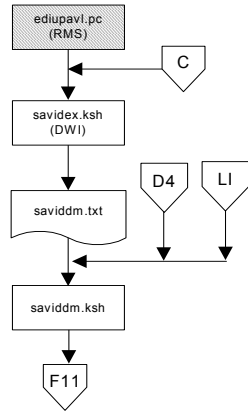


Planning

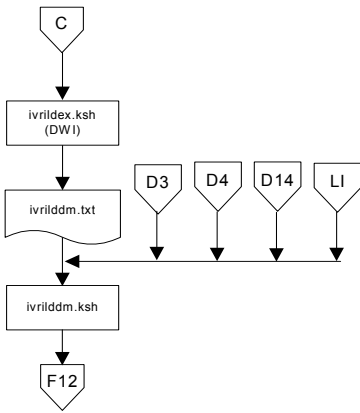


Fact dataflows

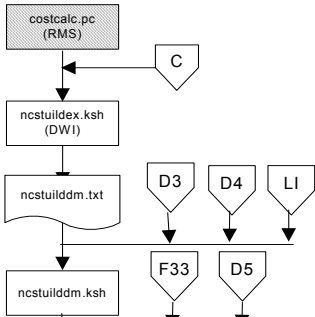
Supplier availability



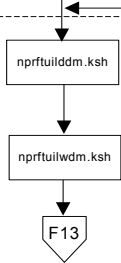
Return to vendor



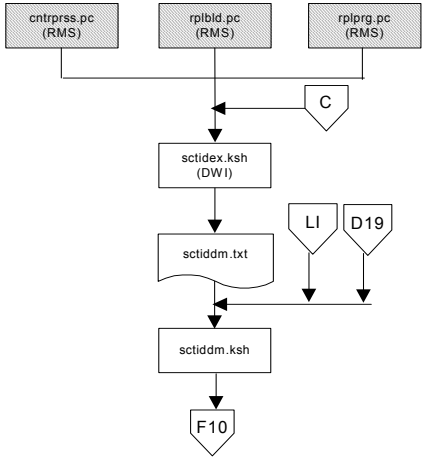
Net cost



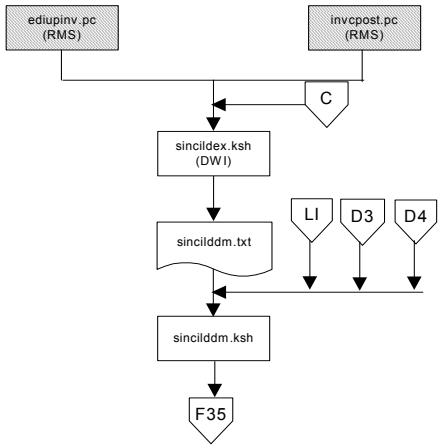
Profit on base cost



Supplier contract

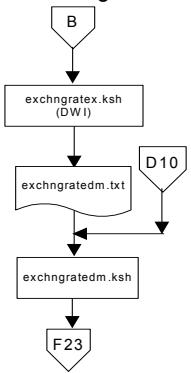


Supplier invoice cost

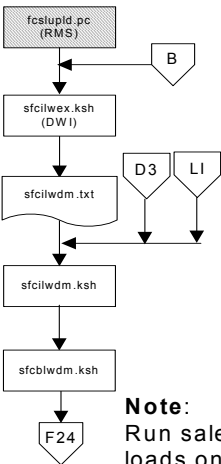


Fact dataflows

Exchange rates

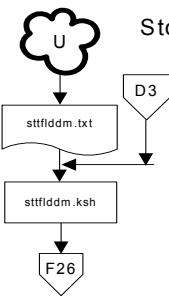


Sales forecasts

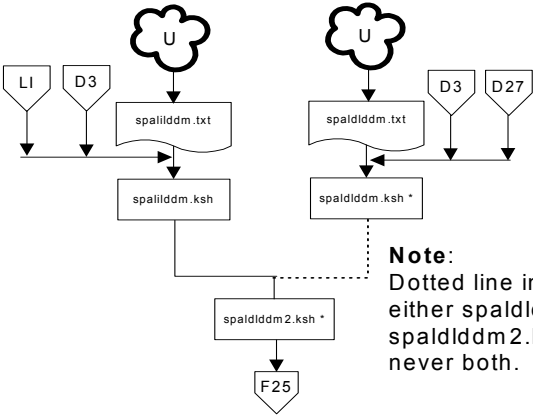


Note:
Run sales forecast fact loads once weekly.

Store traffic

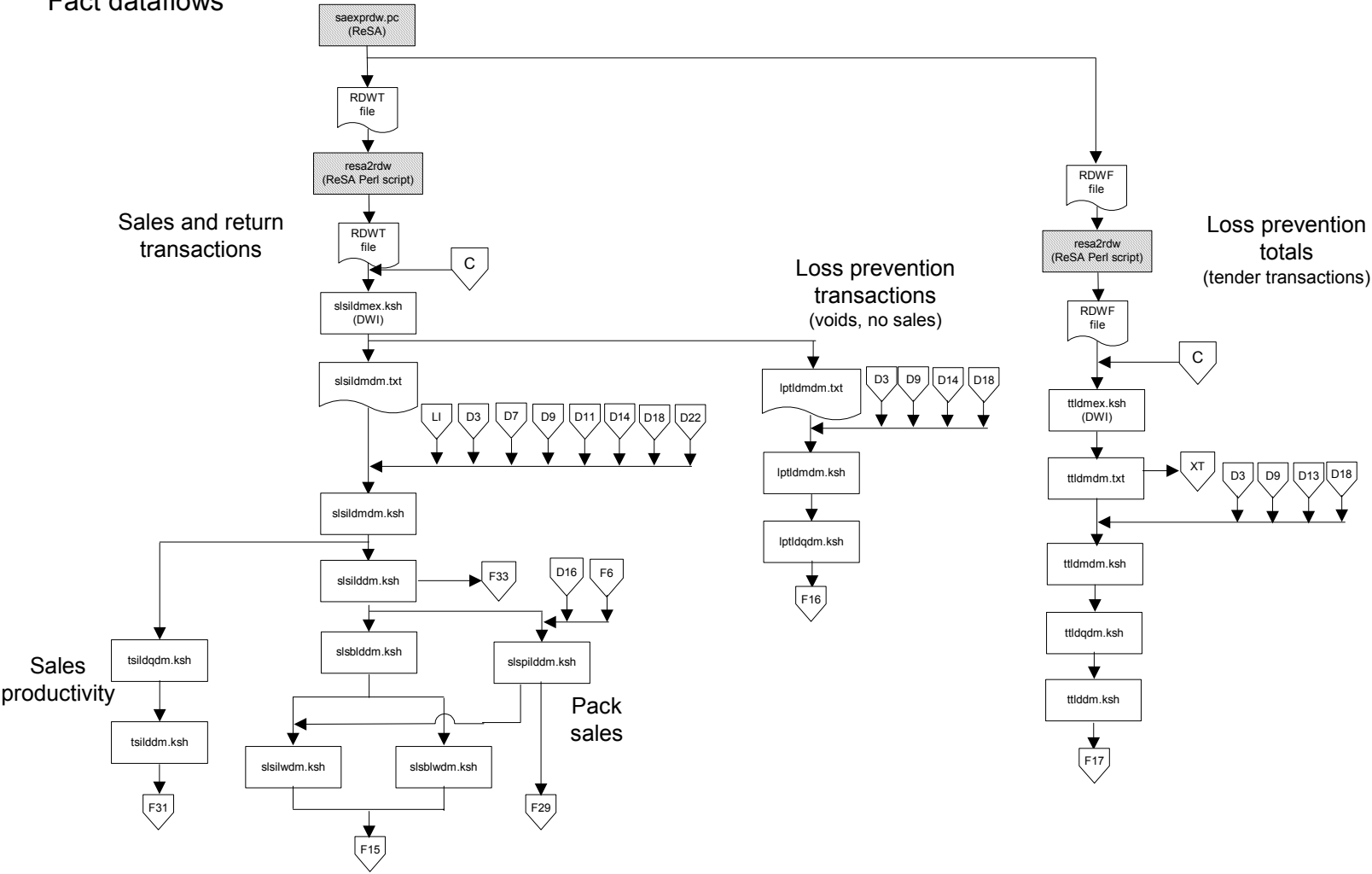


Space allocation

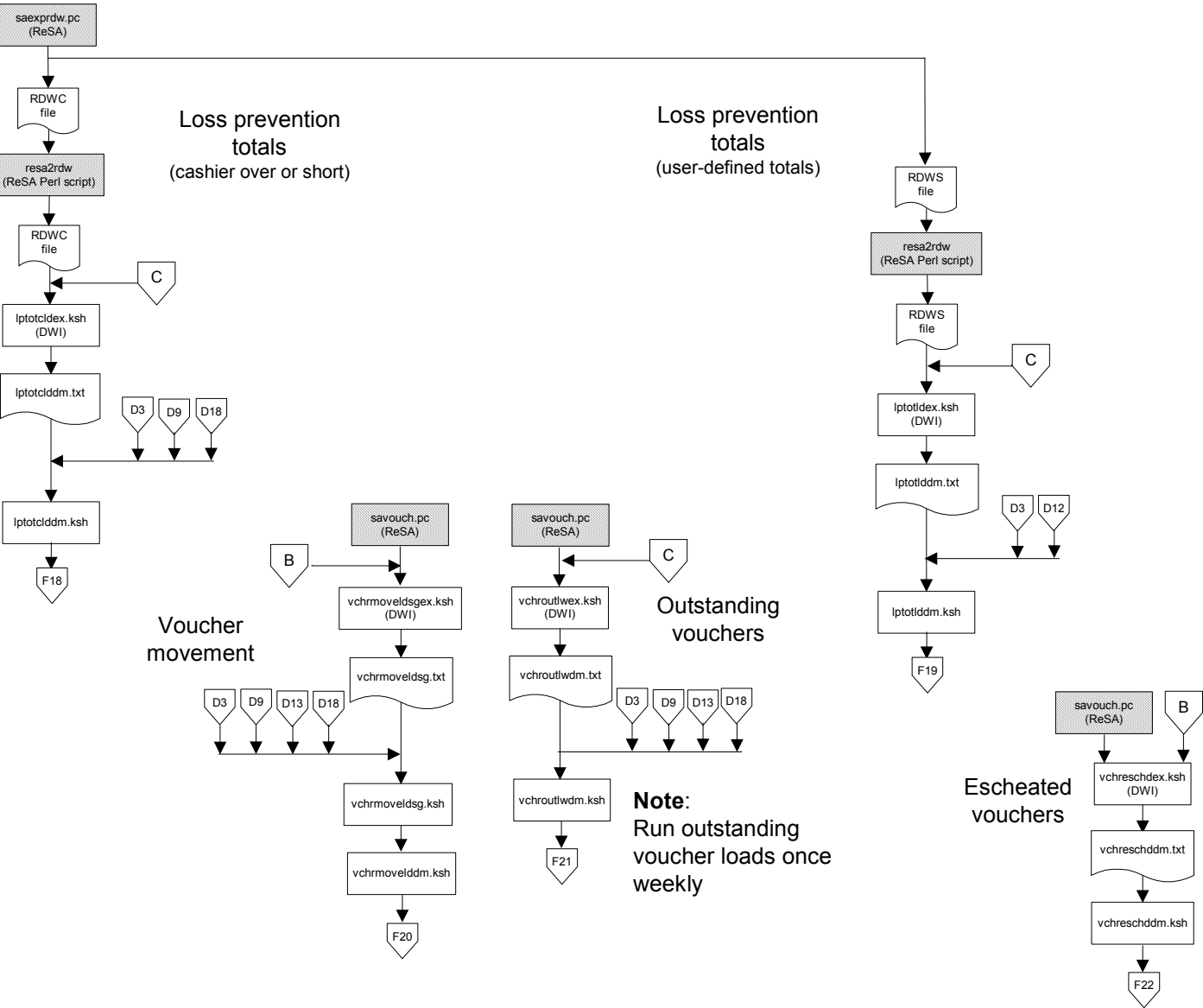


Note:
Dotted line indicates that either spallddm.ksh or spallddm2.ksh runs, never both.

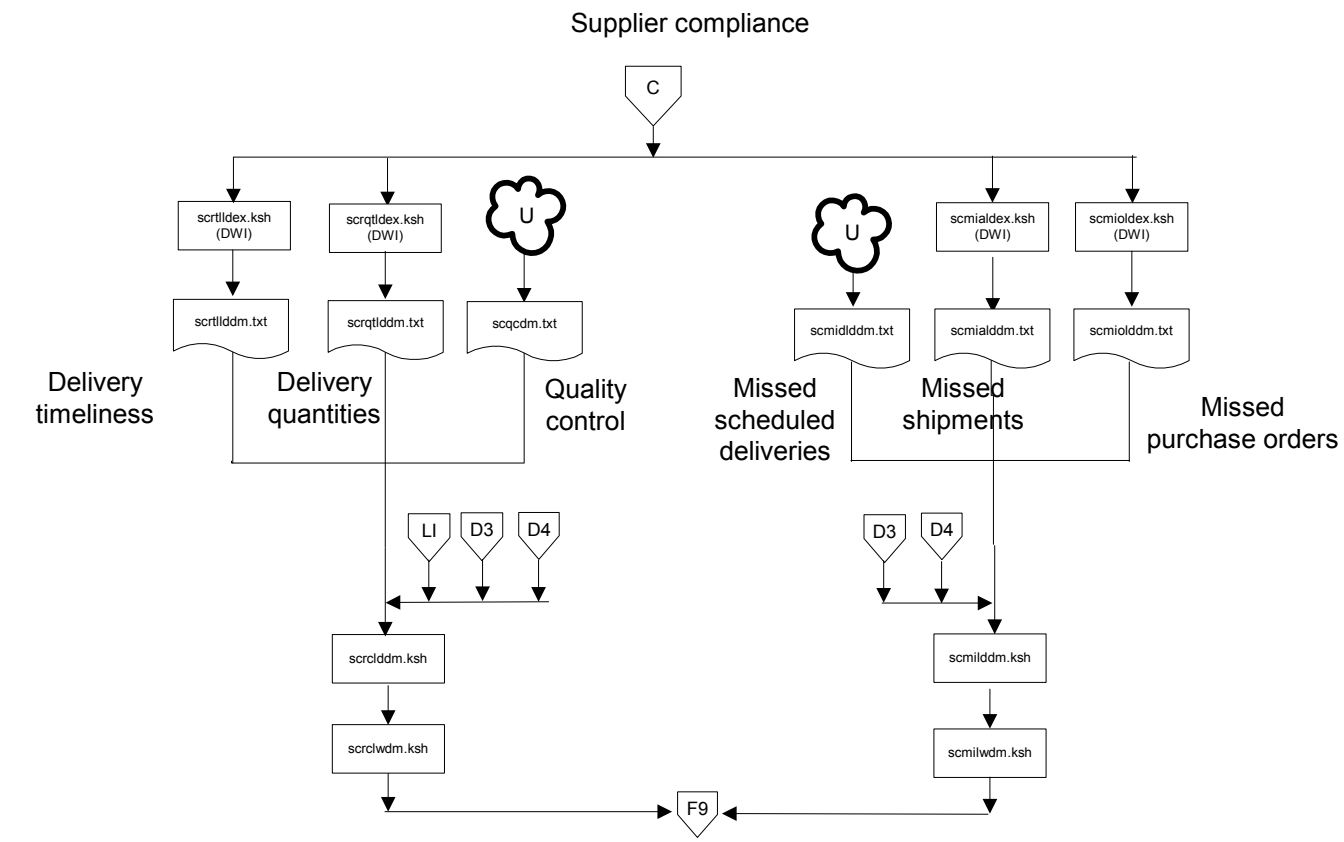
Fact dataflows



Fact dataflows



Fact dataflows



Chapter 8 – Program reference lists

This chapter serves as a reference to the following RDW programs and reference information:

- Dimension extraction and load (RETL Korn shell scripts)
- Fact extraction and load (RETL Korn shell scripts)
- Maintenance (RETL Korn shell scripts)
- The PROGRAM_CONTROL_DM values listed in the tables that follow are explained in a table at the end of this chapter.

By reviewing Chapter 7, “Program flow diagrams”, along with this chapter and Appendix A, “API flat file specifications”, the client should be able to track, down to the table and column level, all the fact and dimension data that flows into RDW.

Dimension programs

When referencing the tables below, note the following:

The dimension DM KSH modules do not have an “argument” column in the following table because these modules do not require a path/file_name parameter. Dimension modules assume source text files will be located in \$MMHOME/data and named <DM KSH module name>.txt. If clients wish to change this default path, they will need to pass in their own path/file_name at the command line.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
cmptrdm.ksh	Competitor Dimension	Dimension Load		cmptrdm.txt	cmptrdm.schema	CMPTR_DM	DIM_TOP	UPDATE	
cmptrex.ksh	Competitor Dimension	Dimension Extraction	RMS	COMPETITOR	cmptrdm.schema	cmptrdm.txt			
cmptrlmdm.ksh	Competitor Dimension	Dimension Load		cmptrlmdm.txt	cmptrlmdm.schema	CMPTR_LOC_MTX_DM	DIM_MTX	INSERT	
cmptrlmex.ksh	Competitor Dimension	Dimension Extraction	RMS	COMP_STORE_LINK, CODE_DETAIL	cmptrlmdm.schema	cmptrlmdm.txt			

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
cmptrlocdm.ksh	Competitor Dimension	Dimension Load		cmptrlocdm.txt	cmptrlocdm.schema	CMPTR_LOC_DM	DIM_LOW	UPDATE	
cmptrlocex.ksh	Competitor Dimension	Dimension Extraction	RMS	COMP_STORE	cmptrlocdm.schema	cmptrlocdm.txt			
crnycddm.ksh	Currency Code Dimension	Dimension Load		crnycddm.txt	crnycddm.schema	CRNCY_CDE_DM	DIM_TOP	UPDATE	
crnycdex.ksh	Currency Code Dimension	Dimension Extraction	RMS	CURRENCIES	crnycddm.schema	crnycddm.txt			
custacctdm.ksh	Customer Account Dimension	Dimension Load	See notes	custacctdm.txt	custacctdm.schema	CUST_ACCNT_TYPE_DM, CUST_ACCNT_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
custclstrdm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	custclstrdm.txt	custclstrdm.schema	CUST_CLSTR_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
custclstrimdm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	custclstrimdm.txt	custclstrimdm.schema	CUST_CLSTR_ITEM_MTX_DM	DIM_MTX	INSERT	Source file supplied by client.
custclupddm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	custclstrdm.txt	custclstrdm.schema	CUST_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
custdemogdm.ksh	Customer and Customer Demographics Dimension	Dimension Load		CUST_DM		CUST_MARITAL_DM, CUST_GENDER_DM, CUST_ETHNIC_DM, CUST_DT_OF_BIRTH_DM, CUST_INCOME_DM, CUST_CHILD_DM, CUST_HH_DM	DIM_STANDALONE	UPDATE	
custdm.ksh	Customer and Customer Demographics Dimension	Dimension Load	RCOM	custdm.txt	custdm.schema	CUST_DM	DIM_TOP	UPDATE_L	
emplydm.ksh	Employee Dimension	Dimension Load		emplydm.txt	emplydm.schema	EMPLY_DM	DIM_TOP	UPDATE	
emplyex.ksh	Employee Dimension	Dimension Extraction	RMS	SA_EMPLOYEE	emplydm.schema	emplydm.txt			
geocdedm.ksh	Customer Geographic Dimension	Dimension Load	See notes	geocdedm.txt	geocdedm.schema	GEO_CDE_DM	DIM_TOP	UPDATE	Source file supplied by client.
itmclstremdm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	itmclstremdm.txt	itmclstremdm.schema	ITEM_CLSTR_CUST_MTX_DM	DIM_MTX	INSERT	Source file supplied by client.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
itmlkupdm.ksh	Product Dimension	Dimension Lookup		PROD_ITEM_DM, PROD_ITEM_SEASN_MTX_DM		ITEM_KEY_LKUP_TEMP	DIM_LKUP	INSERT	Builds item lookup temp table daily to aid performance of fact loads.
maralmdm.ksh	Market Data Dimension	Dimension Load	See notes	maralmdm.txt	maralmdm.schema	MKT_AREA_LOC_MTX_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
maralvldm.ksh	Market Data Dimension	Dimension Load	See notes	maralvldm.txt	maralvldm.schema	MKT_AREA_LEVEL1_DM, MKT_AREA_LEVEL2_DM, MKT_AREA_LEVEL3_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
mdepdm.ksh	Market Data Dimension	Dimension Load	See notes	mdepdm.txt	mdepdm.schema	MKT_PROD_DEPT_DM, MKT_PROD_DEPT_MTX_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
mitmdm.ksh	Market Data Dimension	Dimension Load	See Notes	mitmdm.txt	mitmdm.schema	MKT_PROD_ITEM_DM, MKT_PROD_ITEM_MTX_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
orgaradm.ksh	Organization Dimension	Dimension Load		orgaradm.txt	orgaradm.schema	ORG_AREA_DM	DIM_LOW	UPDATE	
orgaraex.ksh	Organization Dimension	Dimension Extraction	RMS	AREA	orgaradm.schema	orgaradm.txt			
orgchandm.ksh	Organization Dimension	Dimension Load		orgchandm.txt	orgchandm.schema	ORG_CHANNEL_DM	DIM_MTX	INSERT	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
orgchanex.ksh	Organization Dimension	Dimension Extraction	RMS	CHANNELS	orgchandm.schema	orgchandm.txt			
orgchndm.ksh	Organization Dimension	Dimension Load		orgchndm.txt	orgchndm.schema	ORG_CHAIN_DM	DIM_LOW	UPDATE	
orgchnex.ksh	Organization Dimension	Dimension Extraction	RMS	CHAIN, COMPHEAD	orgchndm.schema	orgchndm.txt			
orgdisdm.ksh	Organization Dimension	Dimension Load		orgdisdm.txt	orgdisdm.schema	ORG_DISTT_DM	DIM_LOW	UPDATE	
orgdisex.ksh	Organization Dimension	Dimension Extraction	RMS	DISTRICT	orgdisdm.schema	orgdisdm.txt			
orgllmdm.ksh	Organization Dimension	Dimension Load		orgllmdm.txt	orgllmdm.schema	ORG_LOCLST_MTX_DM	DIM_MTX	INSERT	
orgllmex.ksh	Organization Dimension	Dimension Extraction	RMS	LOC_LIST_DETAIL	orgllmdm.schema	orgllmdm.txt			
orglocdm.ksh	Organization Dimension	Dimension Load		orglocdm.txt	orglocdm.schema	ORG_LOC_DM	DIM_LOW	UPDATE	
orglocex.ksh	Organization Dimension	Dimension Extraction	RMS	STORE, DISTRICT, CURRENCIES, COUNTRY, STORE_ATTRIBUTES, STORE_FORMAT, STATE, TSFZONE, PROMOZONE, WH, SYSTEM_OPTIONS, WH_ATTRIBUTES, PROMO_ZONE	orglocdm.schema	orglocdm.txt			

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
orgloldm.ksh	Organization Dimension	Dimension Load		orgloldm.txt	orgloldm.schema	ORG_LOCLST_DM	DIM_TOP_F	UPDATE_D	
orglolex.ksh	Organization Dimension	Dimension Extraction	RMS	LOC_LIST_HEAD	orgloldm.schema	orgloldm.txt			
orgltmdm.ksh	Organization Dimension	Dimension Load		orgltmdm.txt	orgltmdm.schema	ORG_LOC_TRAIT_MTX_DM	DIM_MTX	INSERT	
orgltmex.ksh	Organization Dimension	Dimension Extraction	RMS	LOC_TRAITS_MATRIX	orgltmdm.schema	orgltmdm.txt			
orgltrdm.ksh	Organization Dimension	Dimension Load		orgltrdm.txt	orgltrdm.schema	ORG_LOC_TRAIT_DM	DIM_TOP_IDNT'	UPDATE	
orgltrex.ksh	Organization Dimension	Dimension Extraction	RMS	LOC_TRAITS	orgltrdm.schema	orgltrdm.txt			
orgrgndm.ksh	Organization Dimension	Dimension Load		orgrgndm.txt	orgrgndm.schema	ORG_REGN_DM	DIM_LOW	UPDATE	
orgrgnex.ksh	Organization Dimension	Dimension Extraction	RMS	REGION	orgrgndm.schema	orgrgndm.txt			
phasdm.ksh	Product Season Dimension	Dimension Load		phasdm.txt	phasdm.schema	PHASE_DM	DIM_LOW	UPDATE	
phases.ksh	Product Season Dimension	Dimension Extraction	RMS	PHASES	phasdm.schema	phasdm.txt			

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
plnsendm.ksh	Plan Season Dimension	Dimension Load	See notes	plnsendm.txt	plnsendm.schema	PLN_SEASN_DM, TIME_PLN_STD_BY_WK_DM, PLN_SEASN_WK_MTX_DM	DIM_TOP_F	UPDATE_DL	Source file supplied by client.
prdclsdm.ksh	Product Dimension	Dimension Load		prdclsdm.txt	prdclsdm.schema	PROD_CLASS_DM	DIM_LOW	UPDATE	
prdclex.ksh	Product Dimension	Dimension Extraction	RMS	CLASS, MERCHANT, BUYER, DEPS	prdclsdm.schema	prdclsdm.txt			
prdcmpdm.ksh	Company Dimension	Dimension Load		prdcmpdm.txt	prdcmpdm.schema	CMPY_DM	DIM_TOP	UPDATE_L	
prdcmpex.ksh	Company Dimension	Dimension Extraction	RMS	COMPHEAD	prdcmpdm.schema	prdcmpdm.txt			
prddepdm.ksh	Product Dimension	Dimension Load		prddepdm.txt	prddepdm.schema	PROD_DEPT_DM	DIM_LOW	UPDATE	
prddepex.ksh	Product Dimension	Dimension Extraction	RMS	DEPS, CODE_DETAIL, MERCHANT, BUYER	prddepdm.schema	prddepdm.txt			
prddiffdm.ksh	Product Dimension	Dimension Load		prddiffdm.txt	prddiffdm.schema	PROD_DIFF_DM	DIM_STANDALONE	UPDATE	
prddiffex.ksh	Product Dimension	Dimension Extraction	RMS	DIFF_IDS	prddiffdm.schema	prddiffdm.txt			

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
prddivdm.ksh	Product Dimension	Dimension Load		prddivdm.txt	prddivdm.schema	PROD_DIV_DM	DIM_LOW	UPDATE	
prddivex.ksh	Product Dimension	Dimension Extraction	RMS	DIVISION, COMPHEAD, MERCHANT, BUYER	prddivdm.schema	prddivdm.txt			
prddtypdm.ksh	Product Dimension	Dimension Load		prddtypdm.txt	prddtypdm.schema	PROD_DIFF_TYPE_DM	DIM_STANDALONE	UPDATE	<p>1. No more than 30 DIFF types can exist between the text file and RDW. See Appendix A, “Application programming interface (API) flat file specifications”, for more information.</p> <p>2. For more information about DIFF type processing and RDW’s front end, see the RDW 10.2 Middle Tier Installation Guide.</p>
prddtypex.ksh	Product Dimension	Dimension Extraction	RMS	DIFF_TYPES	prddtypdm.schema	prddtypdm.txt			
prdgrpdm.ksh	Product Dimension	Dimension Load		prdgrpdm.txt	prdgrpdm.schema	PROD_GRP_DM	DIM_LOW	UPDATE	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
prdgrpex.ksh	Product Dimension	Dimension Extraction	RMS	GROUPS, MERCHANT, BUYER	prdgrpdm.schema	prdgrpdm.txt			
prdhdrdm.ksh	Promotion Dimension	Dimension Load		prdhdrdm.txt	prmhdrdm.schema	PRMTN_HEAD_DM	DIM_LOW	UPDATE	
prdhrex.ksh	Promotion Dimension	Dimension Extraction	RMS	PROMEVENT, PROMHEAD, PERIOD	prmhdrdm.schema	prdhdrdm.txt			
prdisldm.ksh	Item-Supplier-Location Cross Dimension	Dimension Load		prdisldm.txt	prdisldm.schema	PROD_ITEM_SUPP_LOC_DM	DIM_MTX	INSERT	
prdislex.ksh	Item-Supplier-Location Cross Dimension	Dimension Extraction	RMS	ITEM_SUPP_COUNTRY_LOC, ITEM_MASTER, ITEM_SUPP_COUNTRY_DIM, ITEM_SUPP_COUNTRY, ITEM_LOC, ITEM_SUPPLIER	prdisldm.schema	prdisldm.txt			
prditmclstrdm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	prditmclstrdm.txt	prditmclstrdm.schema	PROD_ITEM_CLSTR_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
prditmclupddm.ksh	Customer and Product Clustering Dimension	Dimension Load	See notes	prditmclstrdm.txt	prditmclstrdm.schema	PROD_ITEM_DM	DIM_STANDALONE	UPDATE	Source file supplied by client.
prditmdm.ksh	Product Dimension	Dimension Load		prditmdm.txt	prditmdm.schema	PROD_ITEM_DM	DIM_STANDALONE	UPDATE	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
prditmex.ksh	Product Dimension	Dimension Extraction	RMS	ITEM_MASTER, UOM_CLASS, CODE_DETAIL	prditmdm.schema	prditmdm.txt			
prditlmdm.ksh	Product Dimension	Dimension Load		prditlmdm.txt	prditlmdm.schema	PROD_ITEMLST_DM	DIM_TOP_F	UPDATE_D	
prditmlex.ksh	Product Dimension	Dimension Extraction	RMS	SKULIST_HEAD	prditlmdm.schema	prditlmdm.txt			
prditmlmdm.ksh	Product Dimension	Dimension Load		prditmlmdm.txt	prditmlmdm.schema	PROD_ITEMLST_MTX_DM	DIM_MTX	INSERT	
prditmlmex.ksh	Product Dimension	Dimension Extraction	RMS	SKULIST_DETAIL, ITEM_MASTER	prditmlmdm.schema	prditmlmdm.txt			
prditmltmdm.ksh	Item-Location Trait Cross Dimension	Dimension Load		prditmltmdm.txt	prditmltmdm.schema	PROD_ITEM_LOC_TRAITS_MTX_DM	DIM_MTX	INSERT	
prditmltmex.ksh	Item-Location Trait Cross Dimension	Dimension Extraction	RMS	ITEM_LOC_TRAITS, ITEM_MASTER, CODE_DETAIL	prditmltmdm.schema	prditmltmdm.txt			
prditmsmdm.ksh	Product Dimension	Dimension Load		prditmsmdm.txt	prditmsmdm.schema	PROD_SEASN_ITEM_MTX_DM	DIM_STANDALONE	UPDATE	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
prditmsmex.ksh	Product Dimension	Dimension Extraction	RMS	ITEM_SEASONS, PHASES, ITEM_MASTER	prditmsmdm.schema	prditmsmdm.txt			prditmsmex.ksh extracts the latest season/phase combination for all tracking level and above items. In other words, if item A is attached to season A/phase A and season A/phase B in RMS, and phase B starts after phase A, only season A/phase B will show up in the matrix association to item A.
prditmuddm.ksh	Product Dimension	Dimension Load		prditmuddm.txt	prditmuddm.schema	PROD_ITEM_UDA_DTL_DM	DIM_TOP_F	UPDATE_DL	
prditmuhdm.ksh	Product Dimension	Dimension Load		prditmuhdm.txt	prditmuhdm.schema	PROD_ITEM_UDA_HEAD_DM	DIM_TOP_F	UPDATE_D	
prditmumdm.ksh	Product Dimension	Dimension Load		prditmumdm.txt	prditmumdm.schema	PROD_ITEM_UDA_MTX_DM	DIM_MTX	INSERT	
prdlvldm.ksh	Product Dimension	Dimension Load		PROD_ITEM_DM		PROD_LEVEL1_DM, PROD_LEVEL2_DM, PROD_LEVEL3_DM	DIM_STANDALONE	UPDATE	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
prdpimdm.ksh	Product Dimension	Dimension Load		prdpimdm.txt	prdpimdm.schema	PROD_PACK_ITEM_MTX_DM	DIM_STANDALONE	UPDATE	
prdpimex.ksh	Product Dimension	Dimension Extraction	RMS	PACKITEM_BREAKOUT, ITEM_MASTER	prdpimdm.schema	prdpimdm.txt			
prdsbcdm.ksh	Product Dimension	Dimension Load		prdsbcdm.txt	prdsbcdm.schema	PROD_SBC_DM	DIM_LOW	UPDATE	
prdsbcex.ksh	Product Dimension	Dimension Extraction	RMS	SUBCLASS, DEPS, CLASS, BUYER, MERCHANT	prdsbcdm.schema	prdsbcdm.txt			
prdudaex.ksh	Product Dimension	Dimension Extraction	RMS	ITEM_MASTER, UDA UDA_ITEM_DATE, UDA_ITEM_FF, UDA_VALUES UDA_ITEM_LOV	prditmuhdm.schema, prditmuddm.schema prditmumdm.schema	prditmuhdm.txt prditmuddm.txt prditmumdm.txt			
prmevtdm.ksh	Promotion Dimension	Dimension Load		prmevtdm.txt	prmevtdm.schema	PRMTN_EVENT_DM	DIM_TOP	UPDATE	
prmevtex.ksh	Promotion Dimension	Dimension Extraction	RMS	PROMEVENT, PROMHEAD, PERIOD	prmevtdm.schema	prmevtdm.txt			
prmschdm.ksh	Promotion Dimension	Dimension Load		prmschdm.txt	prmschdm.schema	PRMTN_SCHM_DM	DIM_LOW	UPDATE	
prmschex.ksh	Promotion Dimension	Dimension Extraction	RMS	PROMEVENT, PROMHEAD, PERIOD, PROM_MIX_MATCH_HEAD, PROM_THRESHOLD_HEAD	prmschdm.schema	prmschdm.txt			

Program	Functional Area	Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
regngrpdm.ksh	Regionality Dimension	Dimension Load		regngrpdm.txt	regngrpdm.schema	REGIONALITY_GRP_DM	DIM_TOP_IDNT	UPDATE_D	
regngrpex.ksh	Regionality Dimension	Dimension Extraction	RMS	SEC_GROUP, CODE_DETAIL	regngrpdm.schema	regngrpdm.txt			
regnmtxdm.ksh	Regionality Dimension	Dimension Load		regngrpdm.txt	regnmtxdm.schema	REGIONALITY_MTX_DM	DIM_MTX	INSERT	
regnmtxex.ksh	Regionality Dimension	Dimension Extraction	RMS	REGIONALITY_MATRIX, ITEM_MASTER, ITEM_SUPP_COUNTRY_LOC	regnmtxdm.schema	regnmtxdm.txt			
rgstrdm.ksh	Register Dimension	Dimension Load		ttldmdm.txt, rgstrdm.txt	ttldmdm.schema, rgstrdm.schema	RGSTR_DM	DIM_TOP	INSERT	
rsndm.ksh	Reason Dimension	Dimension Load		rsndm.txt	rsndm.schema	REASN_DM	DIM_TOP	UPDATE	
rsnex.ksh	Reason Dimension	Dimension Extraction	RMS	CODE_DETAIL, INV_ADJ_REASON, INV_STATUS_TYPES, QC_FAILURE_CODES, CODE_HEAD, NON_MERCH_CODE_HEAD	rsndm.schema	rsndm.txt			
seasndm.ksh	Product Season Dimension	Dimension Load		seasndm.txt	seasndm.schema	SEASN_DM, TIME_STD_BY_DAY_DM, TIME_STD_BY_WK_DM	DIM_TOP	UPDATE_L	
seasnex.ksh	Product Season Dimension	Dimension Extraction	RMS	SEASONS	seasndm.schema	seasndm.txt			

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
subtrantypdm.ksh	Sub-Transaction Type Dimension	Dimension Load		subtrantypdm.txt	subtrantypdm.schema	SUB_TRAN_TYPE_DM	DIM_TOP	UPDATE	
subtrantypex.ksh	Sub-Transaction Type Dimension	Dimension Extraction	RMS	CODE_DETAIL	subtrantypdm.schema	subtrantypdm.txt			
supctrdm.ksh	Supplier Dimension	Dimension Load		supctrdm.txt	supctrdm.schema	SUPP_CNTRCT_DM	DIM_LOW	UPDATE	
supctrex.ksh	Supplier Dimension	Dimension Extraction	RMS	CONTRACT_HEADER, CODE_DETAIL	supctrdm.schema	supctrdm.txt			
supsupdm.ksh	Supplier Dimension	Dimension Load		supsupdm.txt	supsupdm.schema	SUPP_DM	DIM_TOP	UPDATE_L	
supsupex.ksh	Supplier Dimension	Dimension Extraction	RMS	SUPS, SUP_ATTRIBUTES, CURRENCIES, SYSTEM_OPTIONS	supsupdm.schema	supsupdm.txt			
suptrmdm.ksh	Supplier Dimension	Dimension Load		suptrmdm.txt	suptrmdm.schema	SUPP_TRAIT_MTX_DM	DIM_MTX	INSERT	
suptrmex.ksh	Supplier Dimension	Dimension Extraction	RMS	SUP_TRAITS_MATRIX	suptrmdm.schema	suptrmdm.txt			
suptrtdm.ksh	Supplier Dimension	Dimension Load		suptrtdm.txt	suptrtdm.schema	SUPP_TRAIT_DM	DIM_TOP_IDNT	UPDATE	
suptrtex.ksh	Supplier Dimension	Dimension Extraction	RMS	SUP_TRAITS	suptrtdm.schema	suptrtdm.txt			
tndrtypedm.ksh	Tender Type Dimension	Dimension Load		tndrtypedm.txt	tndrtypedm.schema	TNDR_TYPE_DM	DIM_TOP	UPDATE	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Notes
tndrtypex.ksh	Tender Type Dimension	Dimension Extraction	RMS	POS_TENDER_TYPE_HEAD, CODE_DETAIL	tndrtypedm.schema	tndrtypedm.txt			
ttltypdm.ksh	ReSA Total Type Dimension	Dimension Load		ttltypdm.txt	ttltypdm.schema	TOTAL_TYPE_DM	DIM_TOP	UPDATE	
ttltypex.ksh	ReSA Total Type Dimension	Dimension Extraction	ReSA/RMS	SA_TOTAL_HEAD	ttltypdm.schema	ttltypdm.txt			

Fact programs

When referencing the tables below, note the following:

- All aggregation modules will derive data from temporary table *_TEMP that is created by the module that loads lowest-level facts from a source system for the fact datamart. For example, slsmkdnilddm.ksh is at item-location-day level. For markdowns, day is the lowest level for time, and week is the next level for time. The lowest level (or base) fact load module, slsmkdnilddm.ksh, needs to create a temp table for the next level aggregation. This temp table will hold today's changes/new facts and will be used by slsmkdnilwddm.ksh to aggregate today's changes to the target week table. The “Source Table or File” column for fact aggregation modules, therefore, is left blank in the program reference list.
- The “Arguments” column lists all the command line parameters that exist in addition to the module name itself.
- For the base fact DM Kornshell modules below, the data file path/file_name is a required command line parameter. The “Arguments” column contains the RDW default data file directory path and file name, such as \$MMHOME/data/cmptrcilddm.txt. If clients wish to change this default path, they will need to substitute their own path/file_name at the command line.
- Unless otherwise noted, the number of days that a transaction can be back posted is limited by the stock ledger. If a transaction is extracted with a date prior to or equal to the last closed end-of-month, then the date will be changed during the extraction to the current business virtual date.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
cmptrpreilddm.ksh	Competitor Pricing	Base Fact with compressed table		cmptrpreilddm.txt	cmptrpreilddm.schema	COMP_PRICING_ITEM_LD_DM, CMPTR_PRICING_IL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/cmptrpreilddm.txt	See Chapter 4 for compressed table and cur table. This module allows backposted data to process to compressed target table.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
cmptrpreildex.ksh	Competitor Pricing	Fact Extraction	RMS	COMP_STORE_LINK, COMP_PRICE_HIST, CURRENCY_RATES, COMP_STORE	cmptrpreilddm.schema	cmptrpreilddm.txt	N/A	N/A	output_file_path/filename	Back posting of competitor pricing data is not limited by the RMS stock ledger. This module will accurately back post competitor pricing facts, regardless of whether the facts occurred before the RMS SYSTEM_VARIABLES.LAST_EOM_DATE.
cstislddm.ksh	Cost	Base Fact with compressed table		cstislddm.txt	cstislddm.schema	COST_ITEM_SUPP_LD_DM, COST_ISL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/cstislddm.txt	See Chapter 4 for compressed table and cur table.
cstislindex.ksh	Cost	Fact Extraction	RMS	PRICE_HIST, ITEM_SUPP_COUNTRY_LOC, ITEM_LOC, ITEM_MASTER	cstislddm.schema	cstislddm.txt	N/A	N/A	output_file_path/filename	
exchngratedm.ksh	Exchange Rates	Base Fact with insert		exchngratedm.txt	exchngratedm.schema	EXCHNG_RATE_CRNCY_DAY_DM	BASEFACT_INS	NSERT	\$MMHOME/data/exchngratedm.txt	Compressed module without cur table.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
exchngratex.ksh	Exchange Rates	Fact Extraction	RMS	CURRENCY_RATES, EURO_EXCHANGE_RATE	exchngratedm.schema	exchngratedm.txt	N/A	N/A	output_file_path/filename	
invblddm.ksh	Inventory Position	Positional Aggregation				INV_SBC_LD_DM	FACT_AGG_POS	UPDATE_G		
invblwdm.ksh	Inventory Position	Positional Aggregation				INV_SBC_LW_DM	FACT_AGG_POS	UPDATE_F		
invilddm.ksh	Inventory Position	Base Fact with compressed table		invilddm.txt	invilddm.schema	INV_ITEM_LD_DM, INV_IL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/invilddm.txt	See Chapter 4 for compressed table and cur table. Inv position cannot be back posted.
invildex.ksh	Inventory Position	Fact Extraction	RMS	ORDLOC_REV,V_PACKS KU_QTY,IF_TRAN_DATA, ITEM_MASTER, ITEM_LOC, ITEM_LOC_SOH, REPL_ITEM_LOC, ORDHEAD, ORDLOC, PACKITEM	invilddm.schema	invilddm.txt	N/A	N/A	out_file_path/filename	
invilwdm.ksh	Inventory Position	Positional Aggregation				INV_ITEM_LW_DM	FACT_AGG_POS	INSERT		
ivailddm.ksh	Inventory Adjustment	Base Fact with incremental update		ivailddm.txt	ivailddm.schema	INV_ADJ_ITEM_LD_DM	BASEFACT_INCR_UPD	UPDATE	\$MMHOME/data/ivailddm.txt	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
ivaildex.ksh	Inventory Adjustment	Fact Extraction	RMS	IF_TRAN_DATA	ivailddm.schema	ivailddm.txt	N/A	N/A	out_file_path/filename	
ivrcplddm.ksh	Inventory Receipts	Aggregation				INV_RCPTS_SBC_LD_DM	FACT_AGG_STD	UPDATE_S		
ivrcplwdm.ksh	Inventory Receipts	Aggregation				INV_RCPTS_SBC_LW_DM	FACT_AGG_STD	UPDATE_F		
ivrcpilddm.ksh	Inventory Receipts	Base Fact with incremental update		ivrcpilddm.txt	ivrcpilddm.schema	INV_RCPTS_ITEM_LD_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/ivrcpilddm.txt	
ivrcpildex.ksh	Inventory Receipts	Fact Extraction	RMS	IF_TRAN_DATA	ivrcpilddm.schema	ivrcpilddm.txt	N/A	N/A	out_file_path/filename	
ivrcpilwdm.ksh	Inventory Receipts	Aggregation				INV_RCPTS_ITEM_LW_DM	FACT_AGG_STD	UPDATE_FS		
ivrilddm.ksh	Return to Vendor	Base Fact with update		ivrilddm.txt	ivrilddm.schema	INV_RTV_SUPP_ITEM_LD_DM	BASEFACT_UPD	UPDATE	\$MMHOME/data/ivrilddm.txt	
ivrildex.ksh	Return to Vendor	Fact Extraction	RMS	RTV_HEAD, RTV_DETAIL, ITEM_LOC	ivrilddm.schema	ivrilddm.txt	N/A	N/A	output_file_path/filename	
ivtblddm.ksh	Inventory Transfers	Aggregation				INV_TSF_SBC_LD_DM	FACT_AGG_STD	UPDATE_S		
ivtblwdm.ksh	Inventory Transfers	Aggregation				INV_TSF_SBC_LW_DM	FACT_AGG_STD	UPDATE_F		

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
ivtilddm.ksh	Inventory Transfers	Base fact with incremental update		ivtilddm.txt	ivtilddm.schema	INV_TSF_ITEM_LD_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/ivtilddm.txt	
ivtildex.ksh	Inventory Transfers	Fact Extraction	RMS	IF_TRAN_DATA	ivtilddm.schema	ivtilddm.txt	N/A	N/A	output_file_path/filename	
ivtilwdm.ksh	Inventory Transfers	Aggregation				INV_TSF_ITEM_LW_DM	FACT_AGG_STD	UPDATE_FS		
ivuilddm.ksh	Unavailable Inventory	Base Fact with update, for compressed table		ivuilddm.txt	ivuilddm.schema	INV_UNAVL_ITEM_LD_DM, INV_UNAVL_IL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/ivuilddm.txt	See Chapter 4 for compressed table and cur table.
ivuidex.ksh	Unavailable Inventory	Fact Extraction	RMS	INV_STATUS_QTY, ITEM_LOC, ITEM_LOC_SOH, IF_TRAN_DATA	ivuilddm.schema	ivuilddm.txt	N/A	N/A	out_file_path/filename	
lptldmdm.ksh	Loss Prevention Transactions(voids, no sales)	Base Fact with incremental update		lptldmdm.txt	lptldmdm.schema	LP_TRAN_LM_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/lptldmdm.txt	Source file comes from DWI module slsildmex.ksh.
lptldqdm.ksh	Loss Prevention Transactions(voids, no sales)	Aggregation				LP_TRAN_LQ_DM	FACT_AGG_STD	UPDATE_S		
lptotclddm.ksh	Loss Prevention Totals (cashier over or short)	Base Fact with incremental update		lptotclddm.txt	lptotclddm.schema	LP_TOT_CSHR_LD_DM	BASEFACT_INCR_UPD	UPDATE	\$MMHOME/data/lptotclddm.txt	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
lptotldex.ksh	Loss Prevention Totals (cashier over or short)	Fact Extraction	ReSA(RDWC file)	RDWC file	<div>Input (formats input data from ReSA): lptotldex.schema</div> <div>Output (formats output text file): lptotlddm.schema</div>	lptotlddm.txt	N/A	N/A	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWC. 2. Before running lptotldex, RDWC input file must have been properly formatted by running ReSA Perl script resa2rdw.
lptotlddm.ksh	Loss Prevention Totals (user defined totals)	Base Fact with incremental update		lptotlddm.txt	lptotlddm.schema	LP_TOT_LD_DM	BASEFACT_INCR_UPD	UPDATE	\$MMHOME/data/lptotlddm.txt	
lptotldex.ksh	Loss Prevention Totals (user defined totals)	Fact Extraction	ReSA(RDWS file)	RDWS file	<div>Input (formats input data from ReSA): lptotldex.schema</div> <div>Output (formats output text file): lptotlddm.schema</div>	lptotlddm.txt	N/A	N/A	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWS. 2. Before running lptotldex, RDWS input file must have been properly formatted by running ReSA Perl script resa2rdw.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
mslsdlwdm.ksh	Market Sales Data	Fact Standalone	See Notes	mslsdlwdm.txt, MKT_PROD_DEPT_DM, TIME_WK_DM, MKT_AREA_LEVEL1_DM, MKT_AREA_LEVEL2_DM, MKT_AREA_LEVEL3_DM	mslsdlwdm.schema	MKT_SLS_DEPT_LEVEL1_W_DM, MKT_SLS_DEPT_LEVEL2_W_DM, MKT_SLS_DEPT_LEVEL3_W_DM	FACT_STANDALONE	UPDATE	\$MMHOME/ data/mslsdlwdm.txt	This module is using fact matrix concept. See Chapter 4 for more details. Source file supplied by client.
mslsilwdm.ksh	Market Sales Data	Fact standalone	See Notes	mslsilwdm.txt, MKT_PROD_ITEM_DM, TIME_WK_DM, MKT_AREA_LEVEL1_DM, MKT_AREA_LEVEL2_DM, MKT_AREA_LEVEL3_DM	mslsilwdm.schema	MKT_SLS_ITEM_LEVEL1_W_DM, MKT_SLS_ITEM_LEVEL2_W_DM, MKT_SLS_ITEM_LEVEL3_W_DM	FACT_STANDALONE	UPDATE	\$MMHOME/ data/mslsilwdm.txt	This module is using fact matrix concept. See Chapter 4 for more details. Source file supplied by client.
ncstuilddm.ksh	Net Cost	Base Fact with update, for compressed table		ncstuilddm.txt	ncstuilddm.schema	NET_COST_SUPP_ITEM_LD_DM, NET_COST_SIL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/ data/ncstuilddm.txt	See Chapter 4 for compressed table and cur table. This module does not allow backposted data.
ncstuilddex.ksh	Net Cost	Fact Extraction	RMS	FUTURE_COST, ITEM_SUPP_COUNTRY, ITEM_LOC	ncstuilddm.schema	ncstuilddm.txt	N/A	N/A	output_file_path/filename	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
nprftuilddm.ksh	Profit on Base Cost	Derivation, see notes		SLS_ITEM_LD_DM, NET_COST_SUPP_ITEM_LD_DM, PROD_ITEM_SUPP_LOC_DM, TIME_DAY_DM		NET_PRFT_SUPP_ITEM_LD_DM	FACT_STANDALONE	UPDATE		This program combines Merchandise Sales data and Net Cost data to provide alternate profit calculations. The facts are not the same as the profit facts on the Sales Transaction tables.
nprftuilwdm.ksh	Profit on Base Cost	Aggregation				NET_PRFT_SUPP_ITEM_LW_DM	FACT_AGG_STD	UPDATE_F		
plcblwdm.ksh	Planning	Base Fact with update	See Notes	plcblwdm.txt	plcblwdm.schema	PLN_CURR_SBC_LW_DM	BASEFACT_UPD	UPDATE	\$MMHOME/data/plcblwdm.txt	See Chapter 6, “RDW interfaces” for more information about the TopPlan interface.
ploblwdm.ksh	Planning	Base Fact with update	See Notes	ploblwdm.txt	ploblwdm.schema	PLN_ORIG_SBC_LW_DM, PLN_CURR_SBC_LW_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/ploblwdm.txt	See Chapter 6, “RDW interfaces” for more information about the TopPlan interface.
prcilddm.ksh	Pricing	Base Fact with update, for compressed tables		prcilddm.txt	prcilddm.schema	PRICING_ITEM_LD_DM, PRICING_IL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/prcilddm.txt	See Chapter 4 for compressed table and cur table.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
preildex.ksh	Pricing	Fact Extraction	RMS	PRICE_HIST, ITEM_MASTER	preilddm.schema	preilddm.txt	N/A	N/A	output_file_path/filename	
saviddm.ksh	Supplier Availability	Base Fact with update for compressed tables		saviddm.txt	saviddm.schema	SUPP_AVAIL_ITEM_DAY_DM, SUPP_AVAIL_I_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/saviddm.txt	See Chapter 4 for compressed table and cur table.
savidex.ksh	Supplier Availability	Fact Extraction	RMS	SUP_AVAIL	saviddm.schema	saviddm.txt	N/A	N/A	output_file_path/filename	
scmialdex.ksh	Supplier Compliance	Fact Extraction	RMS	SHIPMENT, ORDHEAD	scmialddm.schema	scmialddm.txt	N/A	N/A	output_file_path/filename	
scmilddm.ksh	Supplier Compliance	Base Fact with insert		scmidlddm.txt , scmialddm.txt , scmiolddm.txt	scmidlddm.schema, scmialddm.schema, scmiolddm.schema	SCMP_RCPT_MISS_LD_DM	FACT_MATRIX	UPDATE_A	\$MMHOME/data/scmidlddm.txt, \$MMHOME/data/scmialddm.txt, \$MMHOME/data/scmiolddm.txt	
scmilwdm.ksh	Supplier Compliance	Aggregation				SCMP_RCPT_MISS_LW_DM	FACT_AGG_STD	UPDATE_FS		

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
scmioldex.ksh	Supplier Compliance	Fact Extraction	RMS	ORDHEAD, ORDLOC, STORE, WH	scmiolddm.schema	scmiolddm.txt	N/A	N/A	output_file_path/filename	
scrclddm.ksh	Supplier Compliance	Derivation, see notes		scrtllddm.txt, scrqlddm.txt, scqcdm.txt	scrtllddm.schema, scrqlddm.schema, scqcdm.schema	SCMP_RCPT_ITEM_LD_DM	FACT_MATRIX	UPDATE_A	scrtllddm.txt, scrqlddm.txt, scqcdm.txt	This program joins the facts of three datamart tables to create one larger fact table. The file scqcdm.txt will be externally supplied.
scrcldwm.ksh	Supplier Compliance	Aggregation				SCMP_RCPT_ITEM_LW_DM	FACT_AGG_STD	UPDATE_FS		
scrqldex.ksh	Supplier Compliance	Fact Extraction	RMS	SHIPMENT, ORDLOC, SHIPSKU, ORDHEAD, IF_TRAN_DATA, ITEM_MASTER, V_PACKSKU_QTY	scrqlddm.schema	scrqlddm.txt	N/A	N/A	output_file_path/filename	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
scrtlldex.ksh	Supplier Compliance	Fact Extraction	RMS	IF_TRAN_DATA, ORDHEAD, SHIPMENT, WH, SOURCE_DLVRY_SCHED, SOURCE_DLVRY_SCHED_DAYS	scrtlldm.schema	scrtlldm.txt	N/A	N/A	output_file_path/filename	
sctiddm.ksh	Supplier Contract	Base Fact with update for compressed tables		sctiddm.txt	sctiddm.schema	SUPP_CNTRCT_ITEM_DAY_DM, SUPP_CNTRCT_I_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/sctiddm.txt	See Chapter 4 for compressed table and cur table.
sctidex.ksh	Supplier Contract	Fact Extraction	RMS	CONTRACT_HEADER, CONTRACT_DETAIL, CONTRACT_COST, ORDHEAD, ORDLOC, ITEM_MASTER	sctiddm.schema	sctiddm.txt	N/A	N/A	output_file_path/filename	Only DWI module that can extract facts above tracking level (RMS allows contract facts at the tracking level and above, even in the same item family). Item_key can be tracking level or above.
sfcbldwm.ksh	Sales Forecasts	Aggregation		See notes on the top for aggregation.		SLS_FCST_SBC_LW_DM	FACT_AGG_POS	UPDATE_GF		This module runs weekly.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
sfcilwdm.ksh	Sales Forecasts	Base Fact with update		sfcilwdm.txt	sfcilwdm.sch ema	SLS_FCST_ITEM_LW_DM	BASEFACT_UPD	UPDATE_A		This module runs weekly.
sfcilwex.ksh	Sales Forecasts	Fact Extraction	RMS	ITEM_FORECAST, DOMAIN_DEPT, ITEM_MASTER, DOMAIN_CLASS, DOMAIN_SUBCLASS	sfcilwdm.sch ema	sfcilwdm.txt	N/A	N/A	output_file_path/filename	This module runs weekly.
sincilddm.ksh	Supplier Invoice Cost	Base Fact with insert		sincilddm.txt	sincilddm.sch ema	SUPP_INVC_COST_ITEM_LD_DM	BASEFACT_INS	INSERT	\$MMHOME/data/sincilddm.txt	
sincildex.ksh	Supplier Invoice Cost	Fact Extraction	ReIM	INVC_HEAD, INVC_DETAIL, INVC_XREF, SHIPMENT	sincilddm.sch ema	sincilddm.txt	N/A	N/A	output_file_path/filename	
slsblddm.ksh	Sales and Returns Transactions	Aggregation				SLS_SBC_LD_DM	FACT_AGG_STD	UPDATE		
slsblwdm.ksh	Sales and Returns Transactions	Aggregation				SLS_SBC_LW_DM	FACT_AGG_STD	UPDATE_FS		
slsilddm.ksh	Sales and Returns Transactions	Aggregation				SLS_ITEM_LD_DM	FACT_AGG_STD	UPDATE_S		

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
slsildmdm.ksh	Sales and Returns Transactions	Base Fact with incremental update		slsildmdm.txt	slsildmdm.schema	SLS_ITEM_LM_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/slsildmdm.txt	
slsildmex.ksh	Sales and Returns Transactions	Fact Extraction	ReSA(RDWT file)	PROMSKU, PROMSTORE, ITEM_MASTER, PROMDEPT, PROM_THRESHOLD_SKU, PROM_THRESHOLD_DEPT, PROM_MIX_MATCH_BUY, PROM_MIX_MATCH_GET, VAT_ITEM, STORE, ITEM_LOC_SOH, CLASS	Input (formats input data from ReSA): slsildmex.schema Outputs (formats output text files): slsildmdm.schema lptldmdm.schema	lptldmdm.txt, slsildmdm.txt	N/A	N/A	st_sls_out_file_path/st_sls_out_file st_lp_out_file_path/st_lp_out_file in_file_path/in_file	1. This module takes one input file from ReSA (RDWT file), and outputs two flat files: a sales transaction file and a sales transaction loss prevention file. 2. Input file name must begin with RDWT. 3. Before running slsildmex, the RDWT input file must have been properly formatted by running ReSA Perl script resa2rdw.
slsilwdm.ksh	Sales and Returns Transactions	Aggregation				SLS_ITEM_LW_DM	FACT_AGG_STD	UPDATE_F		
slsmkdnilddm.ksh	Markdowns	Base Fact with update		slsmkdnilddm.txt	slsmkdnilddm.schema	SLS_MKDN_ITEM_LD_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/slsmkdnilddm.txt	

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
slsmkdnildex.ksh	Markdowns	Fact Extraction	RMS	IF_TRAN_DATA	slsmkdnilddm.schema	slsmkdnilddm.txt	N/A	N/A	output_file_path/filename with path	
slsmkdnildwm.ksh	Markdowns	Aggregation				SLS_MKDN_ITEM_LW_DM	FACT_AGG_STD	UPDATE_FS		
slsmkdnildpiddm.ksh	Markdowns	Derivation, See notes		SLS_MKDN_ITEM_LD_DM, PRICING_ITEM_LD_DM, PROD_PACK_ITEM_MTX_DM, TIME_DAY_DM		SLS_MKDN_PACK_ITEM_LD_DM	FACT_MATRIX	UPDATE_S		This module blows out sales markdown facts on the SLS_MKDN_ITEM_LD_DM to their respective pack component sales markdown facts.
slspilddm.ksh	Pack Sales	Derivation, see notes		SLS_ITEM_LD_DM, PRICING_ITEM_LD_DM, PROD_PACK_ITEM_MTX_DM, TIME_DAY_DM		SLS_PACK_ITEM_LD_DM	FACT_MATRIX	UPDATE_S		This module selects sales facts for pack items and breaks down the pack items into their component items.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
spaldlddm.ksh	Space Allocation	Base Fact with update, for compressed table	See Notes	spaldlddm.txt	spaldlddm.schema	SPACE_ALLOC_DEPT_LD_DM, SPACE_ALLOC_DL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/spaldlddm.txt	The client either runs spaldlddm.ksh or department space allocation module spaldlddm2.ksh. Source data is supplied by the client. This module will allow backposted data to process to the compressed target table.
spaldlddm2.ksh	Space Allocation	Aggregation				SPACE_ALLOC_DEPT_LD_DM, SPACE_ALLOC_DL_CUR_DM	FACT_STANDALONE	UPDATE		This module aggregates space allocation.
spalilddm.ksh	Space Allocation	Base Fact with update, for compressed table	See Notes	spalilddm.txt	spalilddm.schema	SPACE_ALLOC_ITEM_LD_DM, SPACE_ALLOC_IL_CUR_DM	BASEFACT_UPD	UPDATE_L	\$MMHOME/data/spalilddm.txt	Space allocation item source data supplied by the client. This module will allow backposted data to process to compressed table.
stlblwdm.ksh	Stock Ledger	Base fact with update		stlblwdm.txt	stlblwdm.schema	INV_VAL_SBC_LW_DM	BASEFACT_UPD	UPDATE	\$MMHOME/data/stlblwdm.txt	This module runs weekly.
stlblwex.ksh	Stock Ledger	Fact Extraction	RMS	WEEK_DATA	stlblwdm.schema	stlblwdm.txt	N/A	N/A	output_file_path/filename	This module runs weekly.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
sttflddm.ksh	Store Traffic	Base Fact	See Notes	sttflddm.txt	sttflddm.schema	STORE_TRAF_LD_DM	BASEFACT_UPD	UPDATE	\$MMHOME/data/sttflddm.txt	Source file supplied by client.
tsilddm.ksh	Sales productivity	Aggregation				SLS_LD_DM	FACT_AGG_STD	UPDATE_FS		
tsildqdm.ksh	Sales productivity	Aggregation				SLS_LQ_DM	FACT_AGG_STD	UPDATE		
ttlddm.ksh	Tender Transaction(Loss Prevention)	Aggregation				TNDR_TRAN_LD_DM	FACT_AGG_STD	UPDATE_MS		
ttldmdm.ksh	Tender Transaction(Loss Prevention)	Base Fact with incremental update		ttldmdm.txt	ttldmdm.schema	TNDR_TRAN_LM_DM	BASEFACT_INCR_UPD	UPDATE_A	\$MMHOME/data/ttldmdm.txt	
ttldmex.ksh	Tender Transaction(Loss Prevention)	Fact Extraction	ReSA (RDWF file)	RDWF file	Input (formats input data from ReSA): ttldmex.schema Output (formats output text file): ttldmdm.schema	ttldmdm.txt	N/A	N/A	output_file_path/filename input_file_path/filename	1. Input file name must begin with RDWF. 2. Before running ttldmex, the RDWF input file must have been properly formatted by running ReSA Perl script resa2rdw.
ttldqdm.ksh	Tender Transaction(Loss Prevention)	Aggregation				TNDR_TRAN_LQ_DM	FACT_AGG_STD	UPDATE_MS		

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
vchreschddm.ksh	Escheated Vouchers	Base Fact with incremental update		vchreschlddm.txt	vchreschlddm.schema	VCHR_ESCH_DAY_DM	BASEFACT_INCR_UPD	UPDATE	\$MMHOME/data/vchreschlddm.txt	
vchreschdex.ksh	Escheated Vouchers	Fact Extraction	RMS	SA_VOUCHER	vchreschddm.schema	vchreschddm.txt	N/A	N/A	output_file_path/filename	
vchrmovelddm.ksh	Voucher Movement	Base Fact aggregated from a staging table				VCHR_MOVE_LD_DM	FACT_AGG_STD	UPDATE_F		
vchrmoveldsg.ksh	Voucher Movement	Staging Table		vchrmoveldsg.txt	vchrmoveldsg.schema	VCHR_MOVE_LD_SG	FACT_MATRIX	UPDATE	\$MMHOME/data/vchrmovelddm.txt	This module loads the staging table VCHR_MOVE_LD_SG, which holds voucher movement facts at the individual voucher level. The module also includes code to decrement voucher movement facts when key information on the source record is updated (for instance, changing cashier or store for an existing voucher).

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Program_Control DM.program_type	Program_Control DM.operation_type	Arguments	Notes
vchrmoveldsgex.ksh	Voucher Movement	Fact Extraction	RMS	SA_VOUCHER	vchrmoveldsg.schema	vchrmoveldsg.txt	N/A	N/A	output_file_path/filename	
vchroutlwdm.ksh	Outstanding Vouchers	Base Fact with insert		vchroutlwdm.txt	vchroutlwdm.schema	VCHR_OUT_LW_DM	FACT_MATRIX	INSERT_G	\$MMHOME/data/vchroutlwdm.txt	This module runs weekly.
vchroutlwex.ksh	Outstanding Vouchers	Fact Extraction	RMS	SA_VOUCHER	vchroutlwdm.schema	vchroutlwdm.txt	N/A	N/A	output_file_path/filename	This module runs weekly.

Maintenance programs

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
factclosedm.ksh	Pre-Batch Maintenance	Maintenance		INV_IL_CUR_DM, INV_UNAVL_IL_CUR_DM, COST_ISL_CUR_DM, SPACE_ALLOC_IL_CUR_DM, SPACE_ALLOC_DL_CUR_DM, NET_COST_SIL_CUR_DM, CMPTR_PRICING_IL_CUR_DM, SUPP_AVAIL_I_CUR_DM, SUPP_CNTRCT_I_CUR_DM, PRICING_IL_CUR_DM, INVSBC_LW_DM, PROD_ITEM_DM PROD_ITEM_RECLASS_DM PROD_DEPT_RECLASS_DM ORG_LOC_RECLASS_DM		INV_ITEM_LD_DM, INV_UNAVL_ITEM_LD_DM, COST_ITEM_SUPP_LD_DM, SPACE_ALLOC_ITEM_LD_DM, CMPTR_PRICING_ITEM_LD_DM, PRICING_ITEM_LD_DM, NET_COST_SUPP_ITEM_LD_DM, SUPP_CNTRCT_ITEM_DAY_DM, SPACE_ALLOC_DEPT_LD_DM, INV_IL_CUR_DM, INV_UNAVL_IL_CUR_DM, COST_ISL_CUR_DM, SPACE_ALLOC_IL_CUR_DM, SPACE_ALLOC_DL_CUR_DM, NET_COST_SIL_CUR_DM, CMPTR_PRICING_IL_CUR_DM, SUPP_AVAIL_I_CUR_DM, SUPP_CNTRCT_I_CUR_DM SUPP_AVAIL_ITEM_DAY_DM INV_SBC_LW_DM		This program processes fact records whose items and/or locations and/or departments have closed or been reclassified. It runs at the beginning of a batch cycle (before mt_prime and after factopendm) and inserts stop records into the compressed tables so the decompression views will no longer pick up records whose items/locations/departments have been reclassified or closed. See Chapter 4, “Compression and partitioning” for details on this program.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
factopendm.ksh	Pre-Batch Maintenance	Maintenance		INV_IL_CUR_DM, INV_UNAVL_IL_CUR_DM, COST_ISL_CUR_DM, SPACE_ALLOC_IL_CUR_DM, SPACE_ALLOC_DL_CUR_DM, NET_COST_SIL_CUR_DM, CMPTR_PRICING_IL_CUR_DM, SUPP_AVAIL_I_CUR_DM, SUPP_CNTRCT_I_CUR_DM, PRICING_IL_CUR_DM, PROD_ITEM_RECLASS_DM, PROD_DEPT_RECLASS_DM, ORG_LOC_RECLASS_DM		INV_ITEM_LD_DM, INV_UNAVL_ITEM_LD_DM, COST_ITEM_SUPP_LD_DM, SPACE_ALLOC_ITEM_LD_DM, CMPTR_PRICING_ITEM_LD_DM, PRICING_ITEM_LD_DM, NET_COST_SUPP_ITEM_LD_DM, SUPP_CNTRCT_ITEM_DAY_DM, SPACE_ALLOC_DEPT_LD_DM, INV_IL_CUR_DM, INV_UNAVL_IL_CUR_DM, COST_ISL_CUR_DM, SPACE_ALLOC_IL_CUR_DM, SPACE_ALLOC_DL_CUR_DM, NET_COST_SIL_CUR_DM, CMPTR_PRICING_IL_CUR_DM, SUPP_AVAIL_I_CUR_DM, SUPP_CNTRCT_I_CUR_DM, SUPP_AVAIL_ITEM_DAT_DM		This program runs immediately before factclosedm.ksh. The program inserts new records into compressed tables with the newly reclassified item/location/department keys after a reclassification day. See Chapter 4, “Compression and partitioning” for details on this program.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
mt_prime.ksh	Pre-Batch Maintenance	Maintenance		MAINT_LOAD_DT_DM, TIME_DAY_DM, TIME_WK_DM		datekey.txt, nextdatekey.txt, currdayidnt.txt, nextdayidnt.txt, currwkidnt.txt, wkenddt.txt, nextwkidnt.txt mthidnt.txt, MAINT_LOAD_DT_DM, PROGRAM_STATUS_DM		<p>1. This program increments the processing date (curr_load_dt) by one day. This program will populate all date-related text files within \$etc directory by joining time_day_dm and time_wk_dm.</p> <p>2. The program, mt_prime.ksh, also prepares the batch cycle to be run by updating the PROGRAM_STATUS_DM table to 'ready' for all modules that have a 'completed' status. Any modules that are still in 'error' status from the previous run will have to be manually updated.</p>

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
orapartseed.ksh	Post-Batch Maintenance	Maintenance		cur table		Partitioned, compressed datamart table	table_name cur_table_name table_level	For partitioned, compressed datamarts, this program seeds the first day of a new partition with the current data from the cur_table. Note that this program applies only to Oracle clients. Explanation of arguments: table name refers to name of the target partitioned table; cur_table_name = name of the CUR position table associated with partitioned target table; table_level refers to level of the target partitioned table, either DAY or WEEK. This program must be run on partition seed days (the first day of the partition) and needs to be called once for every CUR table/compressed target table that is partitioned. For example, run once for INV_ITEM_LD_DM, once for INV_ITEM_LW_DM, once for PRICING_ITEM_LD_DM, and so on. See Chapter 4 for more information. If this module is run on a non-partition seed day, this module processes no data and terminates successfully.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
seasonopendm.ksh	Pre-Batch Maintenance	Maintenance		SEASN_DM PROD_SEASN_ITEM_MTX_DM INV_UNAVL_IL_CUR_DM, INV_IL_CUR_DM		INV_UNAVL_ITEM_LD_DM, INV_UNAVL_IL_CUR_DM, INV_IL_CUR_DM, INV_ITEM_LD_DM, INV_ITEM_LW_DM		When an item crosses from one product season to another, a new prod_seasn_key is associated with the item_key. This change needs to be reflected in compressed facts that contain prod_seasn_key, namely Inventory Position. Seasonopendm maintains the inventory position datamart facts when a season change occurs.
pre_dwi_extract.ksh	Pre-DWI maintenance	Maintenance	RMS	PERIOD, SYSTEM_OPTIONS, SYSTEM_VARIABLES, CURRENCY_RATES		class_level_vat_ind.txt, consolidation_code.txt, domain_level.txt, last_eom_date.txt, max_backpost_days.txt, multi_currency_ind.txt, prime_currency_code.txt, prime_exchng_rate.txt, stkldgr_vat_incl_retl_ind.txt, vat_ind.txt, vdate.txt		This module expects these text files to exist in \$MMHOME/rfx/etc when it runs. Text files containing default values for the very first run are included in the installation process.

Program	Functional Area	Module Type	External Data Source	Source Table or File	Schema File	Target File or Table	Arguments	Notes
pre_dwi_temp.ksh	Pre-DWI maintenance	Maintenance	RMS	CURRENCY_RATES, WH, EURO_EXCHANGE_RATE, STORE, SUPS, CONTRACT_HEADER, INVC_DETAIL, INVC_HEAD		curr_tran_day_temp, loc_exchng_rate_temp, supp_exchng_rate_temp, cntrect_exchng_rate_temp, invc_exchng_rate_temp		
post_dwi_temp.ksh	Post-DWI maintenance	Maintenance						This module drops the tables created by pre_dwi_temp.ksh.

Program type and operation type descriptions

With only a few exceptions, every RDW RETL module contains a program type and an operation type. The program type and operation type tell specific dimension and/or fact RDW RETL libraries how to process the data. The following tables detail every program type and operation type combination.

Dimension types

With regard to dimension types, the following assumptions apply:

- All dimension modules need to have a valid program type and operation type to be able to process data correctly.
- Dimension libraries handle much of the data processing by:
 - Creating one or more temporary tables
 - Analyzing the temporary table(s)
 - Creating an index on the temporary table
 - Generating the surrogate key for new and/or major changed records
 - Updating next_key_val on the MAINT_DIM_KEY_DM table
 - Updating for insertions into the target table based on the temporary tables
 - Updating program status to ‘completed’
- Any exceptions to the above are detailed in the program and/or operation type description fields
- In most cases, a temporary table(s) is created to help with dimension processing. This temporary table might be retained for module(s) later in the flow, such as item_key_lkup_temp. The last batch program that uses the temporary table drops the temporary table.

DIM_TOP

Program type	Program type description	Operation type	Operation type description
DIM_TOP	<ul style="list-style-type: none">Used for dimension modules at the top of the hierarchy or not part of a hierarchy (standalone dimension such as currency), which have surrogate keys for all dimensional identifiers and all maintenance columns.Inserts are treated as new records; therefore, surrogate keys and all maintenance fields are generated before being inserted into the ‘insert’ temporary table.Deletes and minor changes are treated as updated records; therefore, some maintenance fields are updated before being inserted into the ‘update’ temporary table.	INSERT (the same as UPDATE)	
		UPDATE	<ul style="list-style-type: none">Both temporary tables will be dropped in dim_top_ksh.
		UPDATE_L	<ul style="list-style-type: none">Both temporary tables will be kept around for the module itself so that the module can do more processing.The module will manually update program status to ‘completed.’

DIM_TOP_F

Program type	Program type description	Operation type	Operation type description
DIM_TOP_F	<ul style="list-style-type: none">Used for dimension modules at the top of the hierarchy or not part of a hierarchy, which have surrogate keys for all dimensional identifiers but not all maintenance columns. In other words, the history will not be kept if the record is deleted from the system.Inserts are treated as new records; therefore, surrogate keys and all maintenance fields are generated before being inserted into the ‘insert’ temporary table.Deletes are treated as deleted records; therefore, they are inserted into the ‘delete’ temporary table.Minor changes are treated as updated records; therefore, some maintenance fields are updated before being inserted into the ‘update’ temporary table.	UPDATE_D	<ul style="list-style-type: none">All three temporary tables will be dropped in the library dim_top.ksh.
		UPDATE_DL	<ul style="list-style-type: none">All three temporary tables will be kept around for the module itself so the module can finish its processing.The module itself will manually update its program status to ‘completed’.

DIM_TOP_IDNT

Program type	Program type description	Operation type	Operation type description
DIM_TOP_IDNT	<ul style="list-style-type: none">Used for dimension modules at the top of the hierarchy or not part of a hierarchy, which do not have surrogate keys for all dimensional identifiers and might not have all maintenance columns.Inserts are treated as new records; therefore, all maintenance fields are generated before being inserted into the 'insert' temporary table.Deletes are treated as deleted records; therefore, they are inserted into the 'delete' temporary table.Minor changes are treated as updated records; therefore, some maintenance fields are updated before being inserted into the 'update' temporary table.	UPDATE	<ul style="list-style-type: none">Does not use the 'delete' temporary table.Both the 'insert' and 'update' temporary tables will be dropped.
		UPDATE_D	<ul style="list-style-type: none">All three temporary tables will be dropped.

DIM_LOW

Program type	Program type description	Operation type	Operation type description
DIM_LOW	<ul style="list-style-type: none">• Used for dimension modules at the lower level of the hierarchy, which have surrogate keys for all dimensional identifiers and all maintenance columns.• Records are joined with parental table(s) to populate the parental information.• Inserts and major changed inserts are treated as new records; therefore, surrogate keys and all maintenance fields are generated before being inserted into the ‘insert’ temporary table.• Deletes, major changes deletes and minor changes are treated as updated records; therefore, some maintenance fields are updated before being inserted into the ‘update’ temporary table.• A reclass temporary table might be created to keep all major changed records if defined in the module. This temporary table will be used by maintenance modules later.	UPDATE	<ul style="list-style-type: none">• The dim_low.ksh library will handle all the processes.
		UPDATE_L	<ul style="list-style-type: none">• Both temporary tables will be kept around for the module itself so that the module can do more processing.• The module will manually update program status to ‘completed.’

DIM_MTX

Program type	Program type description	Operation type	Operation type description
DIM_MTX	<ul style="list-style-type: none">Used for matrix modules that hold a relationship between one dimension and another dimension.	INSERT	<ul style="list-style-type: none">All records on the target table are deleted.All new records from the text file will be assigned corresponding surrogate keys before being inserted into the target table.No temporary table is generated or used.
		UPDATE	<ul style="list-style-type: none">Records from the text file will be assigned corresponding surrogate keys. The records include new inserts and major changed inserts/updates, and are inserted into a temporary table.The target table will be updated based on the temporary table.The temporary table will be dropped.

DIM_LKUP

Program type	Program type description	Operation type	Operation type description
DIM_LKUP	<ul style="list-style-type: none">Used for modules that populate a subset of a dimension table.	INSERT	<ul style="list-style-type: none">This is a default operation type. No processing depends on this operation type.

DIM_STANDALONE

Program type	Program type description	Operation type	Operation type description
DIM_STANDALONE	<ul style="list-style-type: none">Used for modules that do not need to call libraries.	UPDATE	<ul style="list-style-type: none">This is a default operation type. No processing depends on this operation type.

Fact types

With regard to fact types, the following assumptions apply:

- All fact modules need to have a valid program type and operation type to be able to process data correctly
- Fact libraries handle much of the data processing by:
 - Creating one or more temporary tables
 - Analyzing the temporary table(s)
 - Creating an index on the temporary table
 - Updating or inserting into the target table based on temporary table(s)
 - Updating the program status to 'completed'
- Any exceptions to the above are detailed in the program and/or operation type description fields
- In most cases, a temporary table(s) is created to help with fact processing. This temporary table might be kept around for module(s) later in the flow. The last module to use the temporary table drops it.

BASEFACT_INS

Program type	Program type description	Operation type	Operation type description
BASEFACT_INS	<ul style="list-style-type: none">Used for modules that only insert new records.If records come through with changed positional facts compared to the target table’s positional records, the new position will be inserted into the target table with today’s date.	INSERT	<ul style="list-style-type: none">Records are appended directly onto the temporary table.No temporary table is generated or used.
		UPDATE_A	<ul style="list-style-type: none">Records are appended directly onto the temporary table.The temporary table is kept around for use by another module later in the flow.

BASEFACT_UPD

Program type	Program type description	Operation type	Operation type description
BASEFACT_UPD	<ul style="list-style-type: none">Used for modules that insert new records, and/or update the current records.A temporary table is used to hold the current day’s data to be used in the inserts and updates.	UPDATE	<ul style="list-style-type: none">Records are updated from the temporary table to the target table.The temporary table is dropped.
		UPDATE_L	<ul style="list-style-type: none">Records will be inserted into a temporary table.The temporary table is kept around for use by the module itself and another module later in the scheduling flow.The module itself performs updates and inserts based on the temporary table created by the library. It needs to update its program status to ‘completed’ and drops the temporary table if no aggregation is needed later.All compressed day level tables use this operation type.
		UPDATE_A	<ul style="list-style-type: none">Records are updated/inserted from the temporary table to the target table.The temporary table is kept around for use by another module later in the scheduling flow.

BASEFACT_INCR_UPD

Program type	Program type description	Operation type	Operation type description
BASEFACT_INCR_UPD	<ul style="list-style-type: none">Used for modules that insert new records, and incrementally update the existing records.The first temporary table holds current day’s data on the table.The second temporary table holds incremental updates up to current day.	UPDATE	<ul style="list-style-type: none">Records are merged from both temporary tables and updated/inserted into the target table.The first temporary table is dropped.The second temporary table is dropped.
		UPDATE_A	<ul style="list-style-type: none">Records are merged from both temporary tables and updated/inserted into the target table.The first temporary table is kept around for use by another module later in the scheduling flow.The second temporary table is dropped.

FACT_AGG_POS

Program type	Program type description	Operation type	Operation type description
FACT_AGG_POS	<ul style="list-style-type: none">Used for modules that hold positional data for time and aggregates from a lower level to a higher level in the product hierarchy only.A temporary table from the previous module in the aggregation flow is used to hold the current day's data.	INSERT	<ul style="list-style-type: none">Records are updated on the target table based on the temporary table created by the previous module in the aggregation flow.The temporary table will be dropped.
		UPDATE_F	<ul style="list-style-type: none">A temporary table is created by parameters specified by the module.Records are updated on the target table based on the temporary table.The temporary table will be dropped.Any existing temporary tables from previous modules will be dropped.
		UPDATE_G	<ul style="list-style-type: none">A temporary table is created by parameters specified by the module, including the standard aggregation for product hierarchy.Records are updated on the target table based on the temporary table.The temporary table will be kept around for another module in the flow.Any existing temporary table from previous modules will be dropped.
		UPDATE_GF	<ul style="list-style-type: none">A temporary table is created by parameters specified by the module, including the standard aggregation for product hierarchy.Records are updated on the target table based on the temporary table.The temporary table will be dropped.Any existing temporary table from previous modules will be dropped.

FACT_AGG_STD

Program type	Program type description	Operation type	Operation type description
FACT_AGG_STD	<ul style="list-style-type: none">Used for modules that aggregate from a lower level to a higher level in the time and product hierarchy.The first temporary table is created to hold current day's data.The second temporary table is created to hold aggregates from existing data on the target table and today's data.For DB2 clients only, modules of operation type with or without suffix S use different table spaces for the temporary table. If the previous module in the batch schedule uses operation type with suffix S, the module should not use operation type with suffix S, and vice versa. Program type and operation type do not matter in determining whether the operation type should have a suffix S; the order of batch schedule matters.	UPDATE (UPDATE or UPDATE_S)	<ul style="list-style-type: none">Records are updated/inserted into the target table based on the new temporary table.The temporary table from the previous module in the aggregation flow will be kept around for another module in the flow.The temporary table from the current module will be kept around for another module in the scheduling flow.
		UPDATE_F (UPDATE_F or UPDATE_FS)	<ul style="list-style-type: none">Records are updated/inserted into the target table based on the new temporary table.The temporary table from the previous module in the aggregation flow will be dropped.The temporary table from the current module will be dropped.
		UPDATE_M (UPDATE_M or UPDATE_MS)	<ul style="list-style-type: none">Records are updated/inserted into the target table based on the new temporary table.The temporary table from the previous module in the aggregation flow will be dropped.The temporary table from the current module will be kept around for another module later in the flow.

FACT_MATRIX

Program type	Program type description	Operation type	Operation type description
FACT_MATRIX	<ul style="list-style-type: none">Used for modules that require exception code or additional code for calculations, and/or additional non-standard dimensional joins.Temporary table is created based on the parameters specified by the module.For DB2 clients only, modules of operation type with or without suffix S use different table spaces for the temporary table. If the previous module in the batch schedule uses operation type with suffix S, the module should not use operation type with suffix S, and vice versa. Program type and operation type do not matter in determining whether the operation type should have a suffix S; the order of batch schedule matters.	INSERT	<ul style="list-style-type: none">Records are appended directly onto the target table.No temporary table is generated or used.
		INSERT_G	<ul style="list-style-type: none">Records are appended directly onto the target table.Parameters are specified by the module to indicate the grouping/summing fields.No temporary table is generated or used.
		UPDATE (UPDATE or UPDATE_S)	<ul style="list-style-type: none">Records are updated on the target table based on the temporary table.The temporary table will be dropped.Any existing temporary table from previous modules in the aggregation flow will also be dropped.
		UPDATE_A	<ul style="list-style-type: none">All records from the target table will be updated based on that temporary table.The temporary table will be kept around for another module.Any existing temporary table from previous modules in the aggregation flow will also be dropped.

FACT_STANDALONE

Program type	Program type description	Operation type	Operation type description
FACT_STANDALONE	<ul style="list-style-type: none">Used for fact modules that do not need to call any fact libraries.	UPDATE	<ul style="list-style-type: none">This is a default operation. No processing depends on this operation type.

Maintenance types

MAINTENANCE

Program type	Program type description	Operation type	Operation type description
MAINTENANCE	Used for modules that perform maintenance work and only need to call generic libraries.	UPDATE	<ul style="list-style-type: none">This is a default operation type. No processing depends on this operation type.

Appendix A – Application programming interface (API) flat file specifications

This appendix contains APIs that describe the file format specifications for all text files that serve as the interface between source systems and RDW. For example, these APIs control the formatting of the following:

- The fact and dimension data extracted by the DWI code
- The same fact and dimensions data as it is loaded to the RDW side

In addition to providing individual field description and formatting information, the APIs provide basic business rules for the incoming data.

API format

Each API contains a business rules section and a file layout. Some general business rules and standards are common to all APIs. The business rules are used to ensure the integrity of the information held within RDW. In addition, each API contains a list of rules that are specific to that particular API.

File layout

- **Field Name:** Provides the name of the field in the text file.
- **Description:** Provides a brief explanation of the information held in the field.
- **Max Column Length:** Identifies the maximum length possible for a field. A field may not exceed this length.
- **Data Type/Format:** Data type identifies one of three valid data types: character, number, or date:
 - **Character:** Can hold letters (a,b,c...), numbers (1,2,3...), and special characters (\$,#,&...)
 - **Numbers:** Can hold only numbers (1,2,3...)
 - **Date:** Holds a specific year, month, day combination

Any required formatting for a field is conveyed in the Format section. For example, Number(18,4) refers to number precision and scale. The first value is the precision and always matches the maximum column length; the second value is the scale and specifies how many digits exist to the right of the decimal point.

- **Required Field:** Identifies whether the field can hold a null value. This section holds either a 'yes' or a 'no'. A 'yes' signifies the field may not hold a null value. A 'no' signifies the field may, but is not required to, hold a null value.

General business rules and standards common to all APIs

- **Complete ‘snapshot’ of dimension data:**
A majority of RDW’s dimension code requires a complete view of all current dimensional data (irregardless of whether the dimension information has changed) in order to successfully capture the correct data on the target table. If a complete view of the dimensional data is not provided in the text file, invalid or incorrect dimensional data can result. For instance, not including an active item in the prditmdm.txt file causes that item to be closed (as of the extract date) in the data warehouse. When a sale for the item is processed, the fact program will not find a matching ‘active’ dimension record. Therefore, it is essential, unless otherwise noted in each API’s specific business rules section, that a complete snapshot of the dimensional data be provided in each text file.
- **Leading/trailing values:**
Values entered into the text files are the exact values processed and loaded into the datamart tables. Therefore, the values with leading and/or trailing zeros, characters, or nulls are processed as such. RDW does not strip any of these leading or trailing values, unless otherwise noted in the individual API’s business rules section.
- **Delimiters:**

Note: Make sure the delimiter is never part of your data.

- Within dimension text files, each field must be separated by a pipe (|) character, for example a record from prddivdm.txt may look like the following:
- Within facts text files, each field must be separated by a semi-colon character (;). For example a record from exchngratedm.txt may look like the following:

```
1000|1|Homewares|2006|Henry Stubbs|2302|Craig Swanson
```

```
WIS;20010311;1.73527820592648544918
```

See the RETL 10.2 Programmer’s Guide for additional information.

- **End of Record Carriage Return:**
Each record in the text file must be separated by an end of line carriage return. For example, the three records below, in which each record holds four values, should be entered as:

```
1|2|3|4
```

```
5|6|7|8
```

```
9|10|11|12
```

and not as a continuous string of data, such as:

```
1|2|3|4|5|6|7|8|9|10|11|12
```

- **Character format:**
All API’s should contain ASCII text characters only.

Dimensions

Extraction and load

cmptrdm.txt-file specification

Business rules:

- This text file contains competitor information.
- This text file cannot contain duplicate records for the same cmptr_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CMPTR_IDNT	The unique identifier for competitor.	10	Character	Yes
CMPTR_DESC	The description or name of a competitor.	120	Character	No
CMPTR_ADDR	The competitor address.	255	Character	No
CMPTR_CITY_NAME	Competitor city.	120	Character	No
CMPTR_ST_OR_PRVNC_CDE	A code representing a competitor state or province.	3	Character	No
CMPTR_CNTRY_CDE	Competitor country.	10	Character	No

cmptrlmdm.txt- file specification

Business rules:

- This text file defines the associations between location and competitor location.
- This text file cannot contain duplicate records for the same loc_idnt, cmptr_loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
CMPTR_LOC_IDNT	The unique identifier for competitor store.	10	Character	Yes
TARGET_CMPTR_IND	This field identifies the target competitor of a retailer's store. This competitor's retail will be used along with the primary store within a zone when calculating a recommended retail in Price Management. Valid values are: Y, and N.	1	Character	Yes
CMPTR_RANK	This field captures the rank of each competitor store when compared to the other stores.	2	Number(2)	No
DISTANCE	This field captures the distance between the retailer's store and the competitor's store.	4	Number(4)	No
DISTANCE_UOM_CDE	This field captures the unit of measure code the distance is captured in. Valid values are 1 = 'Miles', 2 = 'Kilometers'.	6	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DISTANCE_UOM_DESC	This field captures the unit of measure description the distance is captured in.	120	Character	No

cmptrlocdm.txt-file specification

Business rules:

- This text file contains non-historical information about competitors and their individual locations.
- This text file cannot contain duplicate records for the same cmptr_loc_idnt, cmptr_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CMPTR_LOC_IDNT	The unique identifier for competitor store.	10	Character	Yes
CMPTR_IDNT	The unique identifier for competitor.	10	Character	Yes
CMPTR_LOC_DESC	The description of competitor store.	120	Character	No
CMPTR_LOC_ADDR	The competitor store's address.	255	Character	No
CMPTR_LOC_CITY_NAME	Competitor store city.	120	Character	No
CMPTR_LOC_ST_OR_PRVNC_CDE	Competitor store state.	3	Character	No
CMPTR_LOC_CNTRY_CDE	Competitor store country.	10	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ESTIMATED_VOLUME	This field is assigned to a competitor's location and indicates the competitor's estimated yearly sales volume.	18	Number(18,4)	No
CMPTR_CRNCY_CDE_IDNT	The unique identifier for currency code. For example, USD is the local currency code for US Dollar.	3	Character	Yes

crnycddm.txt-file specification

Business rules:

- This text file contains currency code information.
- This text file cannot contain duplicate records for the same crncy_cde_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CRNCY_CDE_IDNT	The unique identifier for the currency code. For example, USD is the local currency code for US Dollar.	10	Character	Yes
CRNCY_CDE_DESC	The description of the currency code; for example, description for USD=US Dollar.	120	Character	Yes

emptydm.txt-file specification

Business rules:

- This text file contains the employee data.
- This text file cannot contain duplicate records for the same empty_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
EMPTY_IDNT	The unique identifier for employee.	10	Character	Yes
EMPTY_NAME	Employee name.	120	Character	Yes
EMPTY_ROLE	Indicator for the type of position the employee holds: 'C'ashier, 'S'alesperson, 'O'ther.	1	Character	Yes

orgaradm.txt-file specification

Business rules:

- This text file contains areas within a chain.
- This text file cannot contain duplicate records for the same area_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
AREA_IDNT	The unique identifier for the area.	4	Character	Yes
AREA_DESC	The description of the area.	120	Character	No
AREA_MGR_NAME	The name of the manager for the area.	120	Character	No
CHAIN_IDNT	The unique identifier for the chain.	4	Character	Yes

orgchandm.txt-file specification

Business rules:

- This text file contains channels with a company.
- This text file cannot contain duplicate records for the same channel_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CHANNEL_IDNT	The unique identifier for the channel.	4	Character	Yes
CHANNEL_TYPE	A code that specifies the type of channel.	6	Character	No
CHANNEL_DESC	The description of the channel.	120	Character	No

orgchndm.txt-file specification

Business rules:

- This text file contains chains within a company.
- This text file cannot contain duplicate records for the same chain_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CHAIN_IDNT	The unique identifier for the chain.	4	Character	Yes
CMPY_IDNT	The unique identifier for the company.	4	Character	Yes
CHAIN_DESC	The description of the chain.	120	Character	No
CHAIN_MGR_NAME	The name of the manager for the chain.	120	Character	No

orgdisdm.txt-file specification

Business rules:

- This text file contains districts within a region.
- This text file cannot contain duplicate records for the same distt_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DISTT_IDNT	The unique identifier for the district.	4	Character	Yes
DISTT_DESC	The description of the district.	120	Character	Yes
DISTT_MGR_NAME	The name of the manager responsible for the district.	120	Character	No
REGN_IDNT	The unique identifier for the region.	4	Character	Yes

orgllmdm.txt-file specification

Business rules:

- This text file defines the associations between location and location list.
- This text file cannot contain duplicate records for the same loclst_idnt, loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOCLST_IDNT	The unique identifier for a location list.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes

orglocdm.txt-file specification

Business rules:

- This text file contains locations within a district.
- This text file cannot contain duplicate records for the same loc_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
LOC_DESC	The description or name of the store or warehouse.	120	Character	No
LOC_DESC_10	Contains a 10 character abbreviation of the store name.	10	Character	No
LOC_DESC_3	Contains a 3 character abbreviation of the store name.	3	Character	No
LOC_SECND_DESC	The secondary description or name of the store or warehouse.	120	Character	No
LOC_TYPE_DESC	The description of the loc_type_cde that indicates whether the location is a store or warehouse.	120	Character	No
DISTT_IDNT	The unique identifier for the district.	4	Character	Yes
DISTT_DESC	The description of the district.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CRNCY_CDE_IDNT	The unique identifier for currency.	10	Character	No
CRNCY_CDE_DESC	The description of a local currency code (for example, description for USD = US Dollar). It is the description of the store's preferred currency.	120	Character	No
PHY_WH_IDNT	The unique identifier for the physical warehouse that is assigned to the virtual warehouse.	10	Character	No
VIRTUAL_WH_IDNT	The unique identifier for the virtual warehouse.	10	Character	No
STOCKHOLD_IND	This column indicates whether the location can hold stock. In a non-multichannel environment this will always be Y.	1	Character	No
CHANNEL_IDNT	The unique identifier for the channel (in a multi-channel environment) with which the location is associated.	4	Character	No
LOC_ADDR	The street address of the store or warehouse.	255	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_CITY_NAME	The city in which the store or warehouse is located.	120	Character	No
LOC_ST_OR_PRVNC_CDE	The state or province code in which the store or warehouse is located.	7	Character	No
LOC_CNTRY_CDE	The country code in which the store or warehouse is located.	10	Character	No
LOC_CNTRY_DESC	The description or name of the country code in which the store or warehouse is located.	120	Character	No
LOC_PSTL_CDE	The postal code of the store or warehouse.	30	Character	No
LOC_MGR_NAME	The name of the manager responsible for this store. Only valid for the store locations.	120	Character	No
LOC_FMT_CDE	Code that indicates the type of format of the location. Only valid for store locations.	5	Character	No
LOC_TOT_LINEAR_DISTANCE	Holds the total linear selling space of the location.	8	Number(8)	No
LOC_SELLING_AREA	Contains the location's total selling area.	8	Number(8)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_PRMTN_ZNE_CDE	Code that indicates the promotion zone for which this location is a member. Only valid for the store locations.	5	Character	No
LOC_TRNSFR_ZNE_CDE	Code that indicates the transfer zone for which this location is a member. Only valid for the store locations.	5	Character	No
LOC_VAT_REGN	Contains the number of the Value Added Tax region in which this store or warehouse is contained.	4	Number(4)	No
LOC_VAT_INCLUDE_IND	Indicates whether or not Value Added Tax will be included in the retail prices for the store. Valid values are 'Y' or 'N'.	1	Character	No
LOC_MALL_NAME	Contains the name of the mall in which the store is located.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_DEFAULT_WH	Contains the number of the warehouse that may be used as the default for creating cross-dock masks. This determines which stores are associated with or sourced from a warehouse.	10	Character	No
LOC_BREAK_PAC_IND	Indicates whether or not the warehouse is capable of distributing less than the supplier case quantity. Valid values are 'Y' or 'N'.	1	Character	No
LOC_REMODEL_DT	Contains the date on which the store was last remodeled.		Date (YYYYMMDD)	No
LOC_START_DT	Start date for location.		Date (YYYYMMDD)	No
LOC_END_DT	End date for location.		Date (YYYYMMDD)	No
LOC_TOT_AREA	Contains the total area of the location.	8	Number(8)	No
LOC_NO_LOAD_DOCKS	This field is client specific. The definition and use of this field is customizable for each client.	4	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_NO_UNLOAD_DOCKS	This field is client specific. The definition and use of this field is customizable for each client.	4	Character	No
LOC_UPS_DISTT	Code that indicates the UPS district for which this location is a member. Only valid for the store locations.	2	Number(2)	No
LOC_TIME_ZNE	Code that indicates the time zone for which this location is a member. Only valid for the store locations.	10	Character	No
LOC_FASH_LINE_NO	This field is client specific. The definition and use of this field is customizable for each client.	9	Character	No
LOC_COMP_CDE	This field is client specific. The definition and use of this field is customizable for each client.	2	Character	No
LOC_STORE_VOL_CAT	This field is client specific. The definition and use of this field is customizable for each client.	2	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_PAY_CAT	This field is client specific. The definition and use of this field is customizable for each client.	1	Character	No
LOC_ACCT_CLK_ID	This field is client specific. The definition and use of this field is customizable for each client.	3	Character	No
LOC_FMT_DESC	The description or name of the location format code of this location. Only valid for the store locations.	120	Character	No
LOC_ST_OR_PRVNC_DESC	The description or name of the country code in which the store or warehouse is located.	120	Character	No
LOC_TRNSFR_ZNE_DESC	The description or name of the transfer zone code of this location. Only valid for the store locations.	120	Character	No
LOC_PRMTN_ZNE_DESC	The description or name of the promotion zone code of this location. Only valid for the store locations.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
STORE_CLASS	Contains the code letter indicating the class of which the store is a member. Valid values are A through E.	1	Character	No
START_ORDER_DAYS	Contains the number of days before the store open date that the store will begin accepting orders.	3	Character	No
FORECAST_WH_IND	Indicates whether a warehouse is forecastable and is used by non-RDW Retek applications.	1	Character	No

orgloldm.txt-file specification

Business rules:

- This text file contains one record for each location list. A location list is normally used to group locations for reporting purposes.
- This text file cannot contain duplicate records for the same loclst_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	/ FORMAT	REQUIRED FIELD
LOCLST_IDNT	The unique identifier for a location list.	10	Character	Yes
CREATE_ID	Login id of the person who created the location list.	30	Character	Yes
LOCLST_DESC	The description or name of the location list unique identifier.	120	Character	No

orglmdm.txt-file specification

Business rules:

- This text file defines the associations between location and location traits.
- This text file cannot contain duplicate records for the same loc_trait_idnt, loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TRAIT_IDNT	The unique identifier for the location trait. Only valid entries are for the store locations.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes

orgltrdm.txt-file specification

Business rules:

- This text file contains one row for each location trait. Location traits allow locations, stores, to be grouped based on common characteristics across the organization hierarchy.
- This text file cannot contain duplicate records for the same loc_trait_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TRAIT_IDNT	The unique identifier for the location trait. Only valid entries are for the store locations.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TRAIT_DESC	The description or name of the location trait unique identifier.	120	Character	No

orgrgndm.txt-file specification

Business rules:

- This text file contains regions within an area.
- This text file cannot contain duplicate records for the same regn_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REGN_IDNT	The unique identifier for the region.	4	Character	Yes
REGN_DESC	The description or name of the region.	120	Character	No
REGN_MGR_NAME	Contains the name of the manager for the region.	120	Character	No
AREA_IDNT	The unique identifier for the area.	4	Character	Yes

phasdm.txt-file specification

Business rules:

- This text file contains phases. Phases are periods of time within a season. Each day should fall within no more than one phase.
- This text file cannot contain duplicate records for the same phase_idnt, seasn_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SEASN_IDNT	The unique identifier for a season.	3	Character	Yes
PHASE_IDNT	The unique identifier for a phase.	3	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PHASE_START_DT	The beginning date of the phase.		Date (YYYYMMDD)	Yes
PHASE_END_DT	The ending date of the phase.		Date (YYYYMMDD)	Yes
PHASE_DESC	The description of the phase.	120	Character	No

prdcldsm.txt-file specification

Business rules:

- This text file contains classes within a department.
- This text file cannot contain duplicate records for the same dept_idnt, class_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CLASS_IDNT	The unique identifier for a class.	4	Character	Yes
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes
CLASS_DESC	The description of the class.	120	Character	No
CLASS_BUYR_IDNT	The unique identifier for the buyer of the class.	4	Character	No
CLASS_BUYR_NAME	Name of the buyer for this class of products.	120	Character	No
CLASS_MRCH_IDNT	The unique identifier for the merchandiser of the class.	4	Character	No
CLASS_MRCH_NAME	Name of the merchandiser for this class of products.	120	Character	No

prdcmpdm.txt-file specification

Business rules:

- This text file contains company information.
- This text file cannot contain duplicate records for the same cmpy_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CMPY_IDNT	The unique identifier for the company.	4	Character	Yes
CMPY_DESC	The description of the company.	120	Character	No

prdisldm.txt-file specification

Business rules:

- This text file contains records associating tracking level items with locations and suppliers.
- This text file cannot contain duplicate records for the same supp_idnt, item_idnt, loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
SUPP_PRT_NBR	The corresponding supplier's part number.	30	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PRMY_SUPP_IND	Indicator to maintain and track the primary supplier for an item. Y indicates that this is the primary supplier for this item at this location.	1	Character	No
PRESENTATION_METHOD	Describes the packaging (if any) being taken into consideration in the specified dimensions.	6	Character	No
F_SUPP_CASE_QTY	The quantity of the item in an orderable case pack from the primary supplier.	12	Number(12,4)	No

prddepdm.txt-file specification

Business rules:

- This text file contains departments within a group.
- This text file cannot contain duplicate records for the same dept_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes
GRP_IDNT	The unique identifier for a group.	4	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DEPT_DESC	The description of the department.	120	Character	No
DEPT_BUYR_IDNT	The unique identifier for the buyer of the department.	4	Character	No
DEPT_BUYR_NAME	Name of the buyer for this department of products.	120	Character	No
DEPT_MRCH_IDNT	The unique identifier for the merchandiser of the department.	4	Character	No
DEPT_MRCH_NAME	Name of the merchandiser for this department of products.	120	Character	No
PRFT_CALC_TYPE_CDE	The unique code which determines whether profit will be calculated based on cost or retail for the department.	1	Character	No
PRFT_CALC_TYPE_DESC	The description of what method the profit was calculated for the department. Typically, it would be cost or retail.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PURCH_TYPE_CDE	The code that determines which type of stock the items are within this department (that is, normal stock vs. consignment stock).	1	Character	No
PURCH_TYPE_DESC	The description of the type of merchandise within the department (such as normal stock or consignment stock).	120	Character	No
BUD_INT	Contains the budgeted intake percentage. The term is synonymous with markup percent of retail.	12	Number(12,4)	No
BUD_MKUP	The budgeted markup percentage. This term is synonymous with markup percent of cost.	12	Number(12,4)	No
TOTL_MKT_AMT	The total market amount expected for this department.	18	Number(18,4)	No
MKUP_CALC_TYPE_CDE	The code that determines how markup is calculated for the department.	1	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKUP_CALC_TYPE_DESC	The description of the how the markup is calculated for the department.	120	Character	No
OTB_CALC_TYPE_CDE	The code that determines if open to buy is based on cost or retail for the department.	1	Character	No
OTB_CALC_TYPE_DESC	The description of the whether the open to buy is calculated based on cost or retail.	120	Character	No

prddiffdm.txt-file specification

Business rules:

- This text file contains all item differentiator identifiers, along with their associated NRF industry codes.
- This text file cannot contain duplicate records for the same diff_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DIFF_IDNT	The unique identifier for a differentiator.	10	Character	Yes
DIFF_TYPE	The unique identifier for a differentiator type.	6	Character	No
DIFF_DESC	The description of the differentiator.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
INDUSTRY_CDE	A unique number that represents all possible combinations of sizes.	10	Character	No
INDUSTRY_SUBGROUP	A unique number that represents all different color range groups.	10	Character	No

prddivdm.txt-file specification

Business rules:

- This text file contains divisions within a company.
- This text file cannot contain duplicate records for the same div_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DIV_IDNT	The unique identifier for a division.	4	Character	Yes
CMPY_IDNT	The unique identifier for a company.	4	Character	Yes
DIV_DESC	The description of the division.	120	Character	No
DIV_BUYR_IDNT	The unique identifier for the buyer of the division.	4	Character	No
DIV_BUYR_NAME	The name of the buyer for the division.	120	Character	No
DIV_MRCH_IDNT	The unique identifier for the merchandiser of the division.	4	Character	No
DIV_MRCH_NAME	The name of the merchandiser for the division.	120	Character	No

prddtypdm.txt-file specification

Business rules:

- This text file contains differentiator (diff) types.
- This text file cannot contain duplicate records for the same diff_type.
- The maximum number of diff types allowed in RDW is 30. If new diff types (inserts via the text file) plus existing diff types (on the prod_diff_type_dm table) exceeds 30, data processing errors occur.
- For more information regarding the impact of diff type dimension changes to RDW's front end, see the RDW 10.2 Middle Tier Installation Guide.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DIFF_TYPE	Uniquely identifies a differentiator type.	6	Character	Yes
DIFF_TYPE_DESC	The description of the differentiator type.	120	Character	Yes

prdgrpdm.txt-file specification

Business rules:

- This text file contains groups within a division.
- This text file cannot contain duplicate records for the same grp_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
GRP_IDNT	The unique identifier for a group.	4	Character	Yes
DIV_IDNT	The unique identifier for a division.	4	Character	Yes
GRP_DESC	The description of the group.	120	Character	No
GRP_BUYR_IDNT	The unique identifier for the buyer of the group.	4	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
GRP_BUYR_NAME	The name of the buyer for the group.	120	Character	No
GRP_MRCH_IDNT	The unique identifier for the merchandiser of the group.	4	Character	No
GRP_MRCH_NAME	The name of the merchandiser for the group.	120	Character	No

prditmdm.txt-file specification

Business rules:

- This text file contains items within a subclass, class, and department. The combination of subclass, class and department makes an item unique. For example, item 100 cannot be identified by subclass 10, because subclass 10 can belong to different classes, and represent 2 different subclasses. Item 100 belongs to a combination of subclass, class and department.
- This text file cannot contain duplicate records for the same item_idnt.
- The following columns are not utilized by RDW 10.2 but are included in the prditmdm text file for the benefit of other Retek applications.
 - forecast_ind
 - item_aggregate_ind
 - diff_1_aggregate_ind
 - diff_2_aggregate_ind
 - diff_3_aggregate_ind
 - diff_4_aggregate_ind

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LEVEL1_IDNT	The unique identifier for the first level of the item family.	25	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LEVEL2_IDNT	The unique identifier for the second level of the item family.	25	Character	No
LEVEL3_IDNT	The unique identifier for the third level of the item family.	25	Character	No
ITEM_LEVEL	Number indicating which of the three levels the item resides at. Valid values are 1, 2 and 3.	1	Number(1)	Yes
TRAN_LEVEL	Number indicating which of the three levels transactions occurs for the item's family. Valid values are 1, 2 and 3.	1	Number(1)	Yes
DIFF_1	The unique identifier of a differentiator or a differentiator group.	10	Character	No
DIFF_2	The unique identifier of a differentiator or a differentiator group.	10	Character	No
ITEM_AGGREGATE_IND	Item aggregate indicator, which is used by non-RDW Retek applications.	1	Character	No
DIFF_1_AGGREGATE_IND	Diff_1 aggregate indicator, which is used by non-RDW Retek applications.	1	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DIFF_2_AGGREGATE_IND	Diff_2 aggregate indicator, which is used by non-RDW Retek applications.	1	Character	No
DIFF_3_AGGREGATE_IND	Diff_3 aggregate indicator, which is used by non-RDW Retek applications.	1	Character	No
DIFF_4_AGGREGATE_IND	Diff_4 aggregate indicator, which is used by non-RDW Retek applications.	1	Character	No
PACK_IND	The indicator indicates if the item is a pack. Valid values are Y, N.	1	Character	No
PACK_SELLABLE_CDE	The code indicates whether the pack is sellable. A sellable pack is a group of items that is to be sold as one pack, whether the pack arrived as orderable or whether the retailers took it upon themselves to package and sell the items as a pack. An example of this would be shampoo and conditioner put together and sold as a pack.	6	Character	No
PACK_SELLABLE_DESC	The pack sellable description.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PACK_SIMPLE_CDE	The code indicates whether the pack is simple. A simple pack is a group of multiples of one particular item that is to be sold as one pack. An example would be a twelve pack of cola.	6	Character	No
PACK_SIMPLE_DESC	The pack simple description.	120	Character	No
PACK_ORDERABLE_CDE	The code indicates the pack order type: vendor or buyer. A buyer orderable pack is a pack whose contents are specified by the buyer. A vendor orderable pack is a pack that is packaged by the vendor and can only be ordered that way, for example, a twelve pack of cola either by vendor or buyer.	6	Character	No
PACK_ORDERABLE_DESC	The pack order type description.	120	Character	No
PACKAGE_UOM	The unit of measure associated with the package size.	4	Character	No
PACKAGE_SIZE	The size of the product printed on any packaging (for example: 24 ounces).	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SBCLASS_IDNT	The unique identifier for a subclass.	4	Character	Yes
CLASS_IDNT	The unique identifier for a class to which this item belongs in the product hierarchy.	4	Character	Yes
DEPT_IDNT	The unique identifier for a department to which this item belongs in the product hierarchy.	4	Character	Yes
ITEM_DESC	The long description of the item. This description is used throughout the system to help online users identify the item.	255	Character	No
ITEM_SECND_DESC	The secondary description of the item.	255	Character	No
ITEM_SHRT_DESC	The short description of the item. This description is the default for downloading to the point of sale.	30	Character	No
ITEM_NBR_TYPE_CDE	The code specifies what type the item is. Valid values for this field are ITEM, UPC-A, EAN13, ISBN, etc.	6	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_NBR_TYPE_DESC	The description of the ITEM_NBR_TYPE_CDE. Valid values are 'Retek Item Number', 'UPC-A', 'EAN13', 'ISBN', etc.	40	Character	No
STND_UOM_CDE	The unique identifier of the unit of measure (for example: LBS for pounds).	6	Character	No
STND_UOM_DESC	The description of the UOM_CDE for clarity. Example: 'pounds' for LBS.	120	Character	No
FORECAST_IND	The indicator indicates if the item will be interfaced to an external forecasting system. Valid values are Y, N. This column is used by non-RDW Retek applications.	1	Character	Yes

prditmldm.txt-file specification

Business rules:

- This text file contains one row for each item list. An item list is normally used to group items for reporting purpose.
- This text file cannot contain duplicate records for the same itemlst_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEMLIST_IDNT	The unique identifier for an item list.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CREATE_ID	The login ID of the person who created the item list.	30	Character	Yes
ITEMLIST_DESC	The description of the item list unique identifier.	120	Character	No

prditlmdm.txt-file specification

Business rules:

- This text file contains the associations between item list and tracking level item identifiers.
- This text file cannot contain duplicate records for the same itemlst_idnt and item_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEMLIST_IDNT	The unique identifier for an item list.	10	Character	Yes
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes

prditlmdm.txt-file specification

Business rules:

- This text file contains associations among locations, tracking level items, and their location traits.
- This text file cannot contain duplicate records for the same item_idnt, loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LAUNCH_DT	Holds the date that the item should first be sold at the location.		Date (YYYYMMDD)	No
DEPOSIT_CDE	Indicates whether a deposit is associated with this item at the location	6	Character	No
FOOD_STAMP_IND	Indicates whether the item is approved for food stamps at the location.	1	Character	No
REWARD_ELIGIBLE_IND	Indicates whether the item is legally valid for various types of bonus point/award programs at the location.	1	Character	No
NATL_BRAND_COMP_ITEM	Holds the nationally branded item to which you would like to compare the current item.	25	Character	No
STOP_SALE_IND	Indicates that sale of the item should be stopped immediately at the location.	1	Character	No
ELECT_MKT_CLUBS	Holds the code that represents the electronic marketing clubs to which the item belongs at the location.	6	Character	No
STORE_REORDERABLE_IND	Indicates whether the store may re-order the item.	1	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
FULL_PALLET_ITEM_IND	Indicates whether a store must reorder an item in full pallets only.	1	Character	No
DEPOSIT_CDE_DESC	Deposit code description that indicates whether a deposit is associated with this item at the location.	120	Character	No

prditmsmdm.txt -file specification

Business rules:

- This text file contains associations between a tracking level or above item, and a product season/phase.
- This text file cannot contain duplicate records for the same item.

FIELDNAME	DESCRIPTION	MAX COLUMN LENGTH	FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
PROD_SEASN_IDNT	The unique identifier for a product season.	3	Character	Yes
PROD_PHASE_IDNT	The unique identifier for a product phase.	3	Character	Yes

prditmuddm.txt-file specification

Business rules:

- This text file contains the associations between User Defined Attributes (UDA) at the detail level and item identifiers at the tracking level.
- This text file cannot contain duplicate records for the same item_uda_dtl_idnt and item_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_UDA_HEAD_IDNT	The unique identifier for UDA.	5	Character	Yes
ITEM_UDA_DTL_IDNT	The unique identifier for all text, date, or LOV values for a UDA.	256	Character	Yes
ITEM_UDA_DTL_DESC	UDA value, text, or date description.	255	Character	No

prditmuhdm.txt-file specification

Business rules:

- This text file contains distinct user defined attribute (UDA) values.
- This text file cannot contain duplicate records for the same item_uda_head_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_UDA_HEAD_IDNT	The unique identifier for UDA.	5	Character	Yes
ITEM_UDA_TYPE_CDE	Code designating the UDA type: Valid values are: DT=date, LV=list of values, FF=Free form text.	3	Character	Yes
ITEM_UDA_HEAD_DESC	The UDA description.	120	Character	Yes

prditmumdm.txt-file specification

Business rules:

- This text file contains the associations between UDA (User Defined Attributes) at the detail level and item identifiers at the tracking level.
- This text file cannot contain duplicate records for the same item_uda_dtl_idnt and item_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_UDA_HEAD_IDNT	The unique identifier for the UDA.	5	Character	Yes
ITEM_UDA_DTL_IDNT	The unique identifier for all text, date, or list of values (LOV) for a UDA.	256	Character	Yes
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes

prdpimdm.txt-file specification

Business rules:

- This text file contains the associations between packs and their component tracking-level item identifiers.
- This text file cannot contain duplicate records for the same pack_idnt and item_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PACK_IDNT	The unique identifier for pack.	25	Character	Yes
PACK_ITEM_QTY	Total quantity of a unique item within a pack.	12	Number(12,4)	No
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes

prdsbcdm.txt-file specification

Business rules:

- This text file contains a subclass within a class and a department.
- This text file cannot contain duplicate records for the same dept_idnt, class_idnt, subclass_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SBCLASS_IDNT	The unique identifier for a subclass.	4	Character	Yes
CLASS_IDNT	The unique identifier for a class.	4	Character	Yes
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes
SBCLASS_DESC	The description of the subclass.	120	Character	No
SBCLASS_BUYR_IDNT	The unique identifier for the buyer of this subclass of products.	4	Character	No
SBCLASS_BUYR_NAME	The name of the buyer for this subclass of products.	120	Character	No
SBCLASS_MRCH_IDNT	The unique identifier for the merchandiser of this subclass of products.	4	Character	No
SBCLASS_MRCH_NAME	The name of the merchandiser for this subclass of products.	120	Character	No

prmevtdm.txt-file specification

Business rules:

- This text file contains promotion events and related attributes. Events are time periods used to group promotions for analysis.
- This text file cannot contain duplicate records for the same event_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
EVENT_IDNT	The unique identifier for the event.	10	Character	Yes
EVENT_DESC	The description of the promotion event.	255	Character	No
THEME_DESC	The description of the promotion theme for a given event.	120	Character	No

prmhdrdm.txt-file specification

Business rules:

- This text file contains promotion headers and their attributes. Headers define a promotion and its start/end dates.
- This text file cannot contain duplicate records for the same head_idnt.
- All promotion head_idnt records require a beginning and an end date, even if they are 'dummy' values such as 4444-04-04.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
HEAD_IDNT	The unique identifier for the promotion.	10	Character	Yes
EVENT_IDNT	The unique identifier for the event.	10	Character	Yes
HEAD_NAME	The name of the promotion.	120	Character	No
HEAD_DESC	The description of the promotion.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
BEG_DT	The beginning date of the promotion.		Date (YYYYMMDD)	Yes
END_DT	The ending date of the promotion.		Date (YYYYMMDD)	Yes
THEME_DESC	The description of the promotion theme for a given event.	120	Character	No

prmschdm.txt-file specification

Business rules:

- This text file contains multi-promotion discount schemes and their attributes. Schemes describe a particular type of discount within a promotion, for example, “Buy three, get one free”.
- This text file cannot contain duplicate records for the same schm_idnt.
- All promotion schm_idnt records require a beginning and an end date, even if they are ‘dummy’ values such as 4444-04-04.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SCHM_IDNT	The unique identifier for the promotion scheme.	10	Character	Yes
SCHM_TYPE_CDE	Identifies whether the promotion is mix and match, threshold, multi-unit, or standard.	4	Character	Yes
HEAD_IDNT	The unique identifier for the promotion.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SCHM_DESC	The description of the promotion scheme (for example, “Buy 1, Get 1 Free”).	120	Character	No
SCHM_TYPE_DESC	The full identifier of the promotion scheme type (that is, mix and match, threshold, multi-unit, or standard).	120	Character	No
THEME_DESC	The description of the promotion theme.	120	Character	No
BEG_DT	The beginning date of the promotion.		Date (YYYYMMDD)	Yes
END_DT	The ending date of the promotion.		Date (YYYYMMDD)	Yes

regngrpdm.txt-file specification

Business rules:

- This text file contains regionality group information.
- This text file cannot contain duplicate records for the same regionality_grp_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REGIONALITY_GRP_IDNT	The unique identifier for a regionality group.	4	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REGIONALITY_GRP_DESC	The description of the regionality group.	120	Character	No
REGIONALITY_GRP_ROLE_CDE	The role assigned to this regionality group.	6	Character	No
REGIONALITY_GRP_ROLE_DESC	The description of the role for this regionality group.	120	Character	No

regnmtxdm.txt-file specification

Business rules:

- This text file contains the associations among regionality groups, departments, locations, and suppliers.
- This text file cannot contain duplicate records for the same regionality_grp_idnt, loc_idnt, supp_idnt, dept_idnt combinations.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REGIONALITY_GRP_IDNT	The unique identifier for the user group id that has access to the specified elements.	4	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
SUPP_IDNT	The unique identifier for a vendor.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes

rgstrdm.txt-file specification

Business rules:

- This text file contains register information.
- This text file cannot contain duplicate records for the same rgstr_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
RGSTR_IDNT	The unique identifier for the register.	10	Character	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes

rsndm.txt-file specification

Business rules:

- This text file contains the reason class, types, and codes for the reason dimension. The file can hold various kinds of transaction reasons/codes such as inventory adjustment, return-to-vendor, voids, sales etc. The reason class allows definition of the reason, and the corresponding types and codes can also be defined under the class.
- This text file cannot contain duplicate records for the same reasn_code_idnt, reasn_type_idnt, reasn_class_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
REASN_CLASS_IDNT	The unique identifier for reason class.	6	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
REASN_CODE_DESC	The description of a reason code.	120	Character	No
REASN_TYPE_DESC	The description of a reason type.	120	Character	No
REASN_CLASS_DESC	The description of a reason class.	120	Character	No

seasndm.txt-file specification

Business rules:

- This text file contains seasons. Seasons are arbitrary periods of time around which some retailers organize their buying and selling patterns. Each day should fall within no more than one season.
- This text file cannot contain duplicate records for the same `seasn_idnt`.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SEASN_IDNT	The unique identifier for a season.	3	Character	Yes
SEASN_START_DT	The beginning date for the season.		Date (YYYYMMDD)	Yes
SEASN_END_DT	The ending date for the season.		Date (YYYYMMDD)	Yes
SEASN_DESC	The description of the season.	120	Character	No

subtrantypedm.txt-file specification

Business rules:

- This text file contains sub-transaction type records.
- This text file cannot contain duplicate records for the same `sub_tran_type_idnt`.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUB_TRAN_TYPE_IDNT	The unique identifier for the sub-transaction type.	6	Character	Yes
SUB_TRAN_TYPE_DESC	The description of the sub-transaction type.	120	Character	No

supctrdm.txt-file specification

Business rules:

- This text file contains supplier contract information.
- This text file cannot contain duplicate records for the same cntret_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CNTRCT_IDNT	The unique identifier for the contract.	6	Character	Yes
SUPP_IDNT	The unique identifier for the supplier.	10	Character	Yes
STATUS_CDE	The code representing the status for this contract.	1	Character	Yes
CNTRCT_BEG_DT	The starting date for the contract.		Date (YYYYMMDD)	No
CNTRCT_END_DT	The ending date for the contract.		Date (YYYYMMDD)	No
CNTRCT_DIST	The name of the distributor who will collect the merchandise from the supplier and deliver to the retailer.	40	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CNTRCT_SHIP_MTHD_CDE	The code representing the method of shipment associated with the contract.	4	Character	No
CNTRCT_SHIP_MTHD_DESC	The description of the method of shipment associated with the contract.	120	Character	No
STATUS_DESC	The description of the contract status.	120	Character	No

supsupdm.txt-file specification

Business rules:

- This text file contains a record for each supplier, and it holds details of supplier related attributes.
- This text file cannot contain duplicate records for the same supp_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for vendor.	10	Character	Yes
SUPP_DESC	The description or name of vendors name.	120	Character	Yes
SUPP_SECND_DESC	The secondary description or name of the supplier.	120	Character	No
SUPP_QC_RQRD_IND	This column indicates whether or not this supplier's receipts should be Qced.	1	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_PRE_MARK_IND	This column indicates whether the items supplied by this supplier will be pre-marked.	1	Character	No
SUPP_PRE_TICKET_IND	This column indicates if the supplier pre-marks or pre-prices his goods.	1	Character	No
SUPP_STTS_CDE	Code that indicates if the supplier is currently active.	2	Character	No
SUPP_STTS_DESC	The description of the status code.	120	Character	No
SUPP_EDT_IND	This column indicates if the supplier has EDI capabilities.	1	Character	No
SUPP_DOMESTIC_CDE	Supplier's domestic code.	1	Character	No
SUPP_DOMESTIC_DESC	The description of the supplier's domestic code.	120	Character	No
SUPP_CRNCY_CDE	The code representing the currency that the supplier operates under.	3	Character	No
SUPP_CRNCY_DESC	The description of the supplier's currency code.	120	Character	No
SUPP_VMI_IND	This column indicates whether a supplier is vendor managed inventory supplier.	1	Character	No

suptrmdm.txt-file specification

Business rules:

- This text file defines the associations between supplier and supplier trait.

- This text file cannot contain duplicate records for the same supp_trait_idnt, supp_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	FORMAT	REQUIRED FIELD
SUPP_TRAIT_IDNT	The unique identifier for supplier trait.	10	Character	Yes
SUPP_IDNT	The unique identifier for vendor.	10	Character	Yes

suptrtdm.txt-file specification

Business rules:

- This text file contains supplier trait information.
- This text file cannot contain duplicate records for the same supp_trait_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	FORMAT	REQUIRED FIELD
SUPP_TRAIT_IDNT	The unique identifier for the supplier trait.	10	Character	Yes
MAST_SUPP_FLAG	Flag which indicates whether or not this trait is a master supplier trait. Valid values are 'Y' or 'N'.	1	Character	Yes
SUPP_TRAIT_DESC	The description of the supplier trait.	120	Character	No
MAST_SUPP_CDE	If this supplier trait is a master supplier trait, then this field can contain the number of the master supplier.	10	Character	No

time_13.txt-file specification

Business rules:

- This text file contains one row for one month of a fiscal calendar year.
- This text file cannot contain duplicate records for the same year, quarter, and month.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
Year	13-period fiscal calendar year.	4	Number (YYYY)	Yes
Qtr	13-period fiscal quarter; valid values are 1-4.	1	Number (Format: Q)	Yes
Month (period)	13-period fiscal period; valid values are 1-13.	2	Number (Format: MM)	Yes
First day of the month	The Gregorian date; for example, 20020101 for January 1st 2002.		Date (YYYYMMDD)	Yes
Number of weeks	Contains either the number 4 or 5 depending upon whether it is a 4-week or 5-week period.	1	Number(1)	Yes

time_454.txt-file specification

Business rules:

- This text file contains one row for one month of a fiscal calendar year.
- This text file cannot contain duplicate records for the same year and month.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
Year	454 fiscal calendar year.	4	Number (YYYY)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
Month	Month for the fiscal year in the calendar; for example, 1 for January, 12 for December, etc.	2	Number (MM)	Yes
First day of the month	The Gregorian date; for example, 20020101 for January 1st 2002.		Date (YYYYMMDD)	Yes
Number of weeks	Contains either the number 4 or 5 depending upon whether it is a 4-week or a 5-week month.	1	Number(1)	Yes
Month description	Calendar month description (January, February, etc.).	30	Character	Yes

tndrtypedm.txt-file specification

Business rules:

- This text file contains tender types and their parent tender type groups.
- This text file cannot contain duplicate records for the same tndr_type_id_idnt, tndr_type_grp_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TNDR_TYPE_GRP_IDNT	The unique identifier for the tender type group. An example of a tender type group is cash, check, or credit card.	6	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TNDR_TYPE_ID_IDNT	The unique identifier for tender type ID. An example of a tender type ID is Discover Card, Master Card, or Visa.	6	Character	Yes
TNDR_TYPE_GRP_DESC	The description of the tender type group. An example of the description may be Credit Cards, Cash, or Check.	120	Character	No
TNDR_TYPE_ID_DESC	The description of the tender type ID. An example of the ID description may be Master Card, Visa Gold, or American Express Corporate.	120	Character	No
CASH_EQUIV_FLAG	The indicator of the cash equivalence.	1	Character	No

ttltypdm.txt-file specification

Business rules:

- This text file contains user-defined totals.
- This text file cannot contain duplicate records for the same total_type_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TOTAL_TYPE_IDNT	The unique identifier for the total to be reconciled.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TOTAL_TYPE_DESC	The description of the total type.	255	Character	Yes

wkday.txt-file specification

Business rules:

- This text file contains only one record. That record displays the day description of the first day of the week.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
WKDAY_DESC	This day description will represent the first day of the fiscal week; for example, Monday or Sunday. Note that different countries might use a different day as the first day of the week.	120	Character	Yes

Load only

custacctdm.txt-file specification

Business rules:

- This text file contains customer and account number relationships. It allows account numbers to be linked to specific customers. In the case that two customers have the same account, only the primary account holder can be in this file.
- This text file cannot contain duplicate records for the same cust_idnt, acct_nbr, acct_type_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_IDNT	The unique identifier for a customer.	15	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ACCNT_NBR	Customer account number, possibly from a checking, credit card or loyalty card account.	30	Character	Yes
ACCNT_TYPE_IDNT	The unique identifier for an account type.	3	Character	Yes
ACCNT_TYPE_DESC	The description of an account type (for example, checking, VISA, Master Card, etc.).	30	Character	Yes
ACCNT_GRP_IDNT	The unique identifier for an account group.	3	Character	Yes
ACCNT_GRP_DESC	The description of an account group (for example, credit cards, loyalty cards, etc.).	30	Character	Yes

custclstrdm.txt-file specification

Business rules:

- This text file contains all customer clusters and their descriptions. The data must come from an external source.
- This text file cannot contain duplicate records for the same cust_clstr_key.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_CLSTR_KEY	Unique numeric key for a CUST cluster.	4	Number(4)	Yes
CUST_CLSTR_DESC	The description or name of this cluster of customers.	30	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_IDNT	The unique identifier for a customer.	15	Character	Yes

custclstrmdm.txt-file specification

Business rules:

- This text file defines the associations between tracking level items and customer clusters.
- This text file cannot contain duplicate records for the same cust_clstr_key, item_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_CLSTR_KEY	Unique numeric key for a CUST cluster.	4	Number(4)	Yes
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes

custdm.txt-file specification

Business rules:

- This text file contains customer information.
- This text file cannot contain duplicate records for the same cust_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_IDNT	The unique identifier for a customer.	15	Character	Yes
CUST_FIRST_NAME	First name of customer.	120	Character	Yes
CUST_LAST_NAME	Last name of customer.	120	Character	Yes
CUST_MIDDLE_NAME	Middle initial of customer.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_TITLE	A label or heading preceding an individual's name. For example: Mr., Ms., Mrs., Dr.	12	Character	No
CUST_SUFFIX	A label following an individual's name; for example: Jr. or Sr.	12	Character	No
CUST_LAST_NAME_MATERNAL	The last name of the customer's mother.	40	Character	No
CUST_LAST_NAME_PATERNAL	The last name of the customer's father.	40	Character	No
CUST_HOME_ADDR_1	The street address of the customer's home.	30	Character	No
CUST_HOME_ADDR_2	The suite or apartment number of the customer's home.	30	Character	No
CUST_HOME_CITY	The city in which the customer's home is located.	25	Character	No
CUST_HOME_COUNTY	The county in which the customer's home is located.	30	Character	No
CUST_HOME_ST_OR_PRVNC_CDE	The code for the country in which the customer's work is within.	3	Character	No
CUST_HOME_ST_OR_PRVNC_DESC	The state or province in which the customer's home is within.	120	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_HOME_CNTRY_CDE	The code for the country in which the customer's home is located.	10	Character	No
CUST_HOME_PSTL_CDE	The code used by postal service to identify an area in which the customer's home is located.	30	Character	No
CUST_HOME_PSTL_CDE_4	An extension of the postal code used to further narrow an area in which the customer's home is located.	4	Character	No
CUST_WORK_ADDR_1	The street address of the customer's work.	30	Character	No
CUST_WORK_ADDR_2	The suite or apartment number of the customer's work.	30	Character	No
CUST_WORK_CITY	The city in which the customer's workplace is located.	25	Character	No
CUST_WORK_COUNTY	The county in which the customer's workplace is located.	30	Character	No
CUST_WORK_ST_OR_PRVNC_CDE	The code for the state or province in which the customer's work is within.	3	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_WORK_ST_OR_PRVNC_DESC	The state or province in which the customer's work is within.	120	Character	No
CUST_WORK_CNTRY_CDE	The code for the country in which the customer's workplace is located.	10	Character	No
CUST_WORK_PSTL_CDE	The code used by postal service to identify an area in which the customer's workplace is located.	30	Character	No
CUST_WORK_PSTL_CDE_4	An extension of the postal code used to further narrow an area in which the customer's workplace is located.	4	Character	No
CUST_HOME_PHONE	Home phone number for the customer.	30	Character	No
CUST_WORK_PHONE	Work phone number for the customer.	30	Character	No
CUST_FAX	Fax number for the customer.	30	Character	No
CUST_EMAIL	E-mail address for the customer.	80	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_HOME_MAIL_ALLWD_IND	An indicator used to identify whether the customer allows marketing information to be sent to his or her home address.	1	Character	No
CUST_HOME_PHONE_ALLWD_IND	An indicator used to identify if the customer allows marketing information to come to them via home telephone.	1	Character	No
CUST_WORK_MAIL_ALLWD_IND	An indicator used to identify if the customer allows marketing information to be sent to his or her work address.	1	Character	No
CUST_WORK_PHONE_ALLWD_IND	An indicator used to identify if the customer allows marketing information to come to them via work telephone.	1	Character	No
CUST_FAX_ALLWD_IND	An indicator used to identify if the customer allows marketing information to come to them via fax phone.	1	Character	No
CUST_EMAIL_ALLWD_IND	An indicator used to identify if the customer allows marketing information to be sent to his or her electronic mail.	1	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_DT_OF_BIRTH	Customer's date of birth.		Date (YYYYMMDD)	No
CUST_OCCPN	The job which the customer holds.	64	Character	No
CUST_INCOME	Customer's annual income.	18	Number(18,4)	No
CUST_HH_SIZE	Number of people in the household.	2	Number(2)	No
CUST_CHILD_QTY	Number of children the customer has.	2	Number(2)	No
CUST_MARITAL_CDE	A code assigned to a customer to identify his or her marital status.	12	Character	No
CUST_MARITAL_DESC	Customer marital description.	120	Character	No
CUST_GENDER_CDE	A code assigned to a customer to identify his or her gender.	12	Character	No
CUST_GENDER_DESC	Customer's gender description.	120	Character	No
CUST_ETHNIC_CDE	A code assigned to a customer to identify the ethnicity of the customer.	12	Character	No
CUST_ETHNIC_DESC	The ethnicity of the customer.	120	Character	No
CUST_STTS_CDE	A code assigned to a customer to identify the status of a customer.	15	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_STTS_DESC	The status of a customer; for example: active or inactive.	160	Character	No
CUST_TAX_IDNT	The unique identifier given to a customer by a government agency, used for taxing purposes	30	Character	No
CUST_LEGAL_IDNT	The unique identifier given to a customer by a government agency, used to identify the customer's legal identity; for example: a Social Security Number.	20	Character	No
CUST_LEGAL_DESC	Describes the type of legal identity, such as a Social Security Number.	160	Character	No
CUST_ST_IDNT	An identifier given to a customer by a state government agency. Often, this is a driver's license number.	20	Character	No
CUST_TYPE_IDNT	The unique identifier for what kind of customer the customer is.	15	Character	No
CUST_TYPE_DESC	Describes what kind of customer the customer is; for example: employee, distributor, etc.	160	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_EXT_STRAT_IDNT	The unique identifier for how a customer was obtained.	15	Character	No

geocdedm.txt-file specification

Business rules:

- This text file contains the different types of geographical codes.
- This text file cannot contain duplicate records for the same geo_cde_idnt.

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
GEO_CDE_IDNT	The unique identifier for a geographic area.	10	Character	Yes
GEO_CDE_DESC	The description for a geographic area.	30	Character	No
GEO_AGE	Average age for a geographic area.	4	Number(4,1)	No
GEO_ANCESTRY_CDE	Ancestry code for a geographic area.	4	Character	No
GEO_ANCESTRY_CDE_DESC	Ancestry code description.	30	Character	No
GEO_AUTO_AVAIL_NBR	Auto available number.	3	Number(3,1)	No
GEO_COMMUTE_TIME	Average commute time for a geographic area.	5	Number(5,2)	No
GEO_EDU_LVL_CDE	Average education level for a geographic area.	4	Character	No
GEO_EDU_LVL_CDE_DESC	The description of the average education level.	30	Character	No

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
GEO_FAMILY_TYPE_CDE	Family type code for a geographical area.	4	Character	No
GEO_FAMILY_TYPE_CDE_DESC	The description of the family type code.	30	Character	No
GEO_HOME_NBR_ROOMS	Average number of rooms per home.	4	Number(4,1)	No
GEO_HOUSEHOLD_INCOME	Average household income in a geographical area.	15	Number(15)	No
GEO_HOUSING_VALUE	Average house value for a geographic area.	15	Number(15)	No
GEO_INDUSTRY_CDE	Code for the type of industry in a geographical area.	4	Character	No
GEO_INDUSTRY_CDE_DESC	The description of the industry code.	30	Character	No
GEO_MALE_TO_FEMALE_RAT	Male to female ratio for a geographic area.	12	Number(12,4)	No
GEO_PER_CAPITA_INCOME	Per capita income for a geographic area.	15	Number(15)	No
GEO_PERSONS_TOT	Total number of people in a geographical area.	12	Number(12)	No
GEO_POVERTY_TOT	Total number of people in poverty.	9	Number(9)	No

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
GEO_RENT_TO_OWN_RAT	The ratio of number of people who rent to the number of people who own houses.	12	Number(12,4)	No
GEO_RETIREMENT_INCOME	Average retirement income for a geographical area.	15	Number(15)	No
GEO_URBAN_TO_RURAL_RAT	Urban to rural ratio for a geographical area.	12	Number(12,4)	No
GEO_YR_HOME_BUILT	Average year a home was built in a geographic area.	4	Number(4)	No

itmclstrcmdm.txt-file specification

Business rules:

- This text file contains the relationship between customers and item clusters.
- This text file cannot contain duplicate records for the same item_clstr_key, cust_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	FIELD
ITEM_CLSTR_KEY	Unique numeric key for an item cluster.	4	Number(4)	Yes
CUST_IDNT	The unique identifier for a customer.	15	Character	Yes

maralmdm.txt-file specification

Business rules:

- This text file contains the associations between location and market data.
- This text file cannot contain duplicate records for the same loc_idnt, mkt_area_level1_idnt, mkt_area_level2_idnt, and mkt_area_level3_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
MKT_AREA_LEVEL3_IDNT	The unique identifier for a market area level 3.	16	Character	Yes
MKT_AREA_LEVEL2_IDNT	The unique identifier for a market area level 2.	16	Character	Yes
MKT_AREA_LEVEL1_IDNT	The unique identifier for a market area level 1.	16	Character	Yes

maralvldm.txt-file specification

Business rules:

- This text file contains market area level information.
- This text file cannot contain duplicate records for the same mkt_area_level1_idnt, mkt_area_level2_idnt and mkt_area_level3_idnt combination.

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_AREA_LEVEL3_IDNT	The unique identifier for a market area level 3.	16	Character	Yes
MKT_AREA_LEVEL2_IDNT	The unique identifier for a market area level 2.	16	Character	Yes
MKT_AREA_LEVEL1_IDNT	The unique identifier for a market area level 1.	16	Character	Yes
MKT_AREA_LEVEL3_DESC	The description of the market level 3.	30	Character	No

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_AREA_LEVEL2_DESC	The description of the market level 2.	30	Character	No
MKT_AREA_LEVEL1_DESC	The description of the market level 1.	30	Character	No

mdepdm.txt-file specification

Business rules:

- This text file contains market departments.
- This text file cannot contain duplicate records for the same mkt_dept_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	FORMAT	REQUIRED FIELD
MKT_DEPT_IDNT	The unique identifier for a market department.	13	Character	Yes
MKT_DEPT_DESC	The description of a market department.	30	Character	No
OWNED_FLAG_IND	Indicates an owned department.	1	Character	Yes

mitmdm.txt-file specification

Business rules:

- This text file contains market items.
- This text file cannot contain duplicate records for the same mkt_item_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_ITEM_IDNT	The unique identifier for a market item.	25	Character	Yes
MKT_DEPT_IDNT	The unique identifier for a market department.	13	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_ITEM_DESC	The description of a market item.	40	Character	No
MKT_DEPT_DESC	The description of a market department.	30	Character	No
VENDOR_NAME	The vendor/manufacturer of the market item.	30	Character	No
BRAND_NAME	The brand label of the market item.	30	Character	No
FLAVOR_SCENT	The flavor or scent of the market item.	30	Character	No
MKT_ITEM_SIZE	The market item size.	10	Character	No
PROD_TYPE	The product classification.	20	Character	No
PACK_TYPE	The type of packaging of the market item.	20	Character	No
GENERATION_CDE	3 digit identifier that indicates if the UPC has been revised.	3	Character	No
OWNED_FLAG_IND	Indicates an owned item.	1	Character	Yes

plnsendm.txt-file specification

Business rules:

- This text file contains plan seasons.
- This text file cannot contain duplicate records for the same pln_seasn_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PLN_SEASN_IDNT	The unique identifier for a plan season.	6	Character	Yes
PLN_SEASN_START_DT	Plan season start date.		Date (YYYYMMDD)	Yes
PLN_SEASN_END_DT	Plan season end date.		Date (YYYYMMDD)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PLN_SEASN_DESC	The description of the plan season.	30	Character	Yes

prditmclstrdm.txt-file specification

Business rules:

- This text file contains item clusters.
- This text file cannot contain duplicate records for the same item_clstr_key.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_CLSTR_KEY	Unique numeric key for an item cluster.	4	Number(4)	Yes
ITEM_CLSTR_DESC	The description of an item cluster.	30	Character	No
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes

timelastyrbydaylflm.txt-file specification

Business rules:

- This text file contains user-defined relationships between a given day from this year and the matching day from last year. For example, the third Monday of a particular month this year is matched with the third Monday of the same month last year, regardless of the actual date. Another example might be New Year's Eve this year versus New Year's Eve of last year.
- This text file will only be used during installation.
- This text file cannot contain duplicate records for the same day_idnt and last_yr_lfl_day_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_IDNT	The unique numeric representation for a date.	7	Number(7)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LAST_YR_LFL_DAY_IDNT	The unique numeric representation of the day from last year that corresponds to this year's day.	7	Number(7)	Yes

timelastyrbywklfldm.txt-file specification

Business rules:

- This text file contains user-defined relationship between a give week from this year and the corresponding week from last year. For example the week before Easter of this year matched with last year's week before Easter.
- This text file will only be used during installation.
- This text file cannot contain duplicate records for the same wk_idnt and last_yr_lfl_wk_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
WK_IDNT	The unique numeric representation for the week.	6	Number(6)	Yes
LAST_YR_LFL_WK_IDNT	The unique numeric representation of the week from last year that corresponds to this year's week.	6	Number(6)	Yes

vchragebandm.txt file specification

Business rules:

- This text file contains one row for every voucher age band. The voucher age dimension provides a static age band dimension that is used to categorize gift certificates and other vouchers based on their age upon redemption. Each age band is a client-defined range of age, expressed in calendar days. The age of a voucher is used to determine the age band into which it falls.
- Voucher age bands cannot overlap. For example, if voucher age band 1 has a min of 12 and the max is 20, then the next age band must have a min of 21 and a max greater than or equal to 21.
- This text file will only be used during RDW installation.
- This text file cannot contain duplicate records for the same VCHR_AGE_BAND_KEY.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
VCHR_AGE_BAND_KEY	The unique identifier for the age range into which a voucher falls.	6	Number(6)	Yes
VCHR_AGE_BAND_MIN	The minimum age, expressed as a number of calendar days, for a voucher age band. The limits to the age band are inclusive. For example, if the age band min is 12 and the max is 20, then all vouchers of age 12 to 20, inclusive of the limits, belong to this age band.	6	Number	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
VCHR_AGE_BAND_MAX	The maximum age, expressed as a number of calendar days, for a voucher age band. The limits to the age band are inclusive.	6	Number	Yes
VCHR_AGE_BAND_DESC	The description of the voucher age band.	30	Character	No

Facts

Extraction and load

cmptprcildm.txt-file specification

Business rules:

- This text file contains competitors pricing facts for the client location, competitor location, and item combination on a given day. This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, cmptpr_loc_idnt, day_dt combinations.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
CMPTPR_LOC_IDNT	The unique identifier for a competitor store.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_CMPTR_UNIT_RTL_AMT	The competitor's unit retail amount for a particular item. It is stored in primary currency.	18	Number(18,4)	No
F_CMPTR_UNIT_RTL_AMT_LCL	The competitor's unit retail amount for a particular item. It is stored in local currency.	18	Number(18,4)	No
F_CMPTR_MULTI_UNIT_RTL_AMT	The competitor's multi unit retail amount for a particular item. It is stored in primary currency.	18	Number(18,4)	No
F_CMPTR_MULTI_UNIT_RTL_AMT_LCL	The competitor's multi-unit retail amount for a particular item. It is stored in local currency.	18	Number(18,4)	No
RTL_TYPE_CDE	Code that indicates whether the retail type is regular, promotion, or clearance.	2	Character	Yes
OFFER_TYPE_CDE	This non-aggregatable field identifies the offer type code of the competitor's promotional retail. Examples of valid values are 1 = 'Coupon', 2= 'Mailer', etc.	6	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MULTI_UNITS_QTY	This non-aggregatable field identifies the multi units associated with F_CMPTR_UNIT_RTL_AMT for a particular item.	12	Number(12,4)	No

cstislddm.txt-file specification

Business rules:

- This text file contains cost information for an item, supplier, and location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, supp_idnt, and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
SUPP_IDNT	The unique identifier for a vendor.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_BASE_COST_AMT	The cost value in primary currency.	18	Number(18,4)	No
F_BASE_COST_AMT_LCL	The cost value in local currency.	18	Number(18,4)	No

exchngratedm.txt-file specification

Business rules:

- This text file contains currency exchange rate information.
- This text file cannot contain duplicate records for the same crncy_cde_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CRNCY_CDE_IDNT	The unique identifier for the currency code. For example, USD is the local currency code for US Dollar.	10	Character	Yes
DAY_DT	The calendar date for the day the exchange rate became effective.		Date (YYYYMMDD)	Yes
F_EXCHNG_RATE	The current exchange rate.	18	Number(18,4)	Yes

invilddm.txt-file specification

Business rules:

- This text file contains end of day inventory levels and status for an item and location combination on a given day.
- This text file cannot contain duplicate records for the same item_idnt, loc_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
RTL_TYPE_CDE	Code that indicates whether the retail type is regular, promotion, or clearance.	2	Character	Yes
F_I_SOH_QTY	Stock on hand inventory quantity.	12	Number(12,4)	No
F_I_SOH_COST_AMT	Weighted average cost in primary currency times current stock on hand quantity.	18	Number(18,4)	No
F_I_SOH_COST_AMT_LCL	Weighted average cost in local currency times current stock on hand quantity.	18	Number(18,4)	No
F_I_SOH_RTL_AMT	Unit retail amount in primary currency times current stock on hand quantity.	18	Number(18,4)	No
F_I_SOH_RTL_AMT_LCL	Unit retail amount in local currency times current stock on hand quantity.	18	Number(18,4)	No
F_I_ON_ORD_QTY	On order inventory quantity.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_ON_ORD_COST_AMT	On order stock average cost amount in primary currency.	18	Number(18,4)	No
F_I_ON_ORD_COST_AMT_LCL	On order stock average cost amount in local currency.	18	Number(18,4)	No
F_I_ON_ORD_RTL_AMT	On order stock retail amount in primary currency.	18	Number(18,4)	No
F_I_ON_ORD_RTL_AMT_LCL	On order stock retail amount in local currency.	18	Number(18,4)	No
F_I_IN_TRNST_QTY	Inventory in transit quantity.	12	Number(12,4)	No
F_I_IN_TRNST_COST_AMT	Total cost value of inventory in transit in primary currency.	18	Number(18,4)	No
F_I_IN_TRNST_COST_AMT_LCL	Total local cost value of inventory in transit in local currency.	18	Number(18,4)	No
F_I_IN_TRNST_RTL_AMT	Total retail value of inventory in transit in primary currency.	18	Number(18,4)	No
F_I_IN_TRNST_RTL_AMT_LCL	Total retail value of inventory in transit in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_REPL_ACTV_FLAG	Flag to indicate if end date of this record's time period is within the active and inactive dates for replenishment.	1	Character	No
F_I_REPL_CALC_MTHD_CDE	This column holds the replenishment method code value.	2	Character	No
F_I_MIN_SOH_QTY	The minimum stock on hand quantity.	12	Number(12,4)	No
F_I_MIN_SOH_COST_AMT	The minimum stock on hand average cost amount in primary currency.	18	Number(18,4)	No
F_I_MIN_SOH_COST_AMT_LCL	The minimum stock on hand average cost amount in local currency.	18	Number(18,4)	No
F_I_MIN_SOH_RTL_AMT	The minimum stock on hand retail amount in primary currency.	18	Number(18,4)	No
F_I_MIN_SOH_RTL_AMT_LCL	The minimum stock on hand retail amount in local currency.	18	Number(18,4)	No
F_I_MAX_SOH_QTY	The maximum stock on hand quantity.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_MAX_SOH_COST_AMT	The maximum stock on hand average cost amount in primary currency.	18	Number(18,4)	No
F_I_MAX_SOH_COST_AMT_LCL	The maximum stock on hand average cost amount in local currency.	18	Number(18,4)	No
F_I_MAX_SOH_RTL_AMT	The maximum stock on hand retail amount in primary currency.	18	Number(18,4)	No
F_I_MAX_SOH_RTL_AMT_LCL	The maximum stock on hand retail amount in local currency.	18	Number(18,4)	No
F_I_INCR_PCT	This column holds the replenishment incremental percentage or multiple value. This column is used in replenishment calculations.	12	Number(12,4)	No
F_I_COST_AMT	The weighted average cost for stock in primary currency.	18	Number(18,4)	No
F_I_COST_AMT_LCL	The weighted average cost for stock in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_STD_COST_AMT	The cost of the latest item supplied in primary currency. Used to reflect the difference in unit cost if cost method accounting is used.	18	Number(18,4)	No
F_I_STD_COST_AMT_LCL	The cost of the latest item supplied in local currency. Used to reflect the difference in unit cost if cost method accounting is used.	18	Number(18,4)	No
F_I_RTL_AMT	The corporate unit purchase price for stock in primary currency.	18	Number(18,4)	No
F_I_RTL_AMT_LCL	The corporate unit purchase price for stock in local currency.	18	Number(18,4)	No
F_I_AGED_30_60_QTY	This fact is used to record the quantity of inventory that is between 30 and 60 days old at this location on this day.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_AGED_61_90_QTY	This fact is used to record the quantity of inventory that is between 61 and 90 days old at this location on this day.	12	Number(12,4)	No
F_I_AGED_91_120_QTY	This fact is used to record the quantity of inventory that is between 91 and 120 days old at this location on this day.	12	Number(12,4)	No
F_I_AGED_121_QTY	This fact is used to record the quantity of inventory that is 121 days old or older at this location on this day.	12	Number(12,4)	No
F_I_SLS_ADMN_COST_AMT	This fact could be used to store additional sales and administration cost information for this item, location, and day relationship.	18	Number(18,4)	No
F_I_DIST_COST_AMT	This fact could be used to store additional supply chain cost information for this item, location, and day relationship.	18	Number(18,4)	No

ivailddm.txt-file specification

Business rules:

- This text file contains the inventory adjustment data for an item, location, and reason combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, reasn_type_idnt, reasn_cde_idnt and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_I_ADJ_QTY	Quantity of the total stock on hand adjustment.	12	Number(12,4)	No
F_I_ADJ_COST_AMT	The cost amount of total stock on hand adjustment in primary currency.	18	Number(18,4)	No
F_I_ADJ_COST_AMT_LCL	The cost amount of total stock on hand adjustment in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_I_ADJ_RTL_AMT	The retail amount of total stock on hand adjustment in primary currency.	18	Number(18,4)	No
F_I_ADJ_RTL_AMT_LCL	The retail amount of total stock on hand adjustment in local currency.	18	Number(18,4)	No
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes

ivrcpilddm.txt-file specification

Business rules:

- This text file contains inventory receipts for an item and location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_I_RCPTS_QTY	Quantity of inventory receipts.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
F_I_RCPTS_COST_AMT	The receipt cost amount in primary currency.	18	Number(18,4)	No
F_I_RCPTS_COST_AMT_LCL	The receipt cost amount in local currency.	18	Number(18,4)	No
F_I_RCPTS_RTL_AMT	The receipt retail amount in primary currency.	18	Number(18,4)	No
F_I_RCPTS_RTL_AMT_LCL	The receipt retail amount in local currency.	18	Number(18,4)	No

ivrlddm.txt-file specification

Business rules:

- This text file contains data on inventory returned to a supplier for a supplier, item, and location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, supp_idnt, loc_idnt, and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for a vendor.	10	Character	Yes
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_I_RTV_QTY	Quantity of the stock returned to vendor.	12	Number(12,4)	No
F_I_RTV_COST_AMT	Cost of the stock returned to vendor in primary currency.	18	Number(18,4)	No
F_I_RTV_COST_AMT_LCL	Cost of the stock returned to vendor in local currency.	18	Number(18,4)	No
F_I_RTV_RTL_AMT	Retail amount of the stock returned to vendor, in primary currency.	18	Number(18,4)	No
F_I_RTV_RTL_AMT_LCL	Retail amount of the stock returned to vendor, in local currency.	18	Number(18,4)	No
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes

ivtilddm.txt-file specification

Business rules:

- This text file contains inventory transfers for an item, from-location and to-location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, from_loc_idnt and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a target location for the transfer.	10	Character	Yes
FROM_LOC_IDNT	The unique identifier for a source location for the transfer.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_I_TSF_QTY	The quantity transferred.	12	Number(12,4)	No
F_I_TSF_COST_AMT	The transfer cost amount in primary currency.	18	Number(18,4)	No
F_I_TSF_RTL_AMT	The transfer retail amount in primary currency.	18	Number(18,4)	No

ivuilddm.txt-file specification

Business rules:

- This text file contains unavailable inventory for an item, location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt and day_dt combination.

FIELD NAME	DESCRIPTION	COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_I_UNAVL_QTY	Quantity of unavailable inventory.	12	Number(12,4)	No
F_I_UNAVL_COST_AMT	Average cost amount of the unavailable inventory in primary currency.	18	Number(18,4)	No
F_I_UNAVL_COST_AMT_LCL	Average cost amount of the unavailable inventory in local currency.	18	Number(18,4)	No
F_I_UNAVL_RTL_AMT	Retail amount of the unavailable inventory in primary currency.	18	Number(18,4)	No
F_I_UNAVL_RTL_AMT_LCL	Retail amount of the unavailable inventory in local currency.	18	Number(18,4)	No
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes

lptldm.txt-file specification

Business rules:

- This text file contains all the loss prevention transactions at the transaction-location-day-minute level.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TRAN_IDNT	The unique identifier for a transaction.	30	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The business date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
MIN_IDNT	The unique identifier for the minute, made up of the hour_idnt followed by a number 1-60 to indicate the minute of that hour.	4	Number(4) (HH24MI)	Yes
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
CSHR_IDNT	The unique identifier for a cashier.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
RGSTR_IDNT	The unique identifier for a register.	10	Character	Yes
F_LP_AMT	Loss prevention transaction amount, in primary currency.	18	Number(18,4)	No
F_LP_AMT_LCL	Loss prevention transaction amount, in local currency.	18	Number(18,4)	No
F_DISC_COUPON_COUNT	Total count of discount coupons used on one transaction. Discount coupons are issues by the store as opposed to the manufacturer.	16	Number(16,4)	No
F_DISC_COUPON_AMT	Total amount of discount coupons, in primary currency, used on one transaction. Discount coupons are issues by the store as opposed to the manufacturer.	18	Number(18,4)	No
F_DISC_COUPON_AMT_LCL	Total amount of discount coupons, in local currency, used on one transaction. Discount coupons are issues by the store as opposed to the manufacturer.	18	Number(18,4)	No

lptotclddm.txt-file specification

Business rules:

- This text file contains loss prevention over/short totals.
- In each record, either rgstr_idnt or cshr_idnt should be filled with a value and the other field should be -1.
- Amounts will be summed in the target table by cshr_idnt, rgstr_idnt, loc_idnt, and day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CSHR_IDNT	The unique identifier for the cashier.	10	Character	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
RGSTR_IDNT	The unique identifier for the register.	10	Character	Yes
F_DRAWER_OS_AMT	Over/short amount in primary currency.	18	Number(18,4)	No
F_DRAWER_OS_AMT_LCL	Over/short amount in local currency.	18	Number(18,4)	No

lptotlddm.txt-file specification

Business rules:

- This text file contains user-defined loss prevention totals.
- Amounts will be summed in the target table by total type, location, and day.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
TOTAL_TYPE_IDNT	The unique identifier for the type of the total.	10	Character	Yes
F_TOTAL_AMT	The total amount in primary currency.	18	Number(18,4)	No
F_TOTAL_AMT_LCL	The total amount in local currency.	18	Number(18,4)	No

ncstuilddm.txt-file specification

Business rules:

- This text file contains net cost information.
- This text file cannot contain duplicate transactions for the same item_idnt, supp_idnt, loc_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for the item.	25	Character	Yes
SUPP_IDNT	The unique identifier for the supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the day the cost became effective.		Date (YYYYMMDD)	Yes
F_SUPP_BASE_COST_AMT	The supplier base cost of the item in primary currency. It is the initial cost before any deals or discounts are applied.	18	Number(18,4)	No
F_SUPP_BASE_COST_AMT_LCL	The supplier base cost of the item in local currency. It is the initial cost before any deals or discounts are applied.	18	Number(18,4)	No
F_SUPP_NET_COST_AMT	The supplier net cost for the item in primary currency. It is the defined as the base cost minus any deal components that have been applied by the retailer. If no deals or discounts are applied at this level, the supplier net cost = supplier base cost.	18	Number(18,4)	No
F_SUPP_NET_COST_AMT_LCL	The supplier net cost for the item in local currency. It is the defined as the base cost minus any deal components that have been applied by the retailer. If no deals or discounts are applied at this level, the supplier net cost = supplier base cost.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SUPP_NET_NET_COST_AMT	The supplier net net cost of the item in primary currency. It is defined as the net cost minus any deal components designated by a retailer as applicable to the net net cost. If no deals or discounts are applied at this level, the supplier net net cost = supplier net cost.	18	Number(18,4)	No
F_SUPP_NET_NET_COST_AMT_LCL	The supplier net net cost of the item in local currency. It is defined as the net cost minus any deal components designated by a retailer as applicable to the net net cost. If no deals or discounts are applied at this level, the supplier net net cost = supplier net cost.	18	Number(18,4)	No
F_SUPP_DEAD_NET_COST_AMT	The supplier dead net cost of the item in primary currency. It is the final cost after all deals or discounts have been applied. It is defined as the net net cost minus any deal components designated by a retailer as applicable to the dead net cost. If no deals or discounts are applied at this level, the supplier dead net cost = supplier net net cost.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SUPP_DEAD_NET_COST_AMT_LCL	The supplier dead net cost of the item in local currency. It is the final cost after all deals or discounts have been applied. It is defined as the net net cost minus any deal components designated by a retailer as applicable to the dead net cost. If no deals or discounts are applied at this level, the supplier dead net cost = supplier net net cost.	18	Number(18,4)	No

prcilddm.txt-file specification

Business rules:

- This text file contains prices by item and location combination on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CHNG_CDE	Code that indicates the reason for a price change.	2	Character	No
F_MULTI_UNIT_QTY	Number of units that comprise a multi-unit transaction.	12	Number(12,4)	No
F_UNIT_RTL_AMT	Unit value of new retail valuation/price, in primary currency.	18	Number(18,4)	No
F_UNIT_RTL_AMT_LCL	Unit value of new retail valuation/price, in local currency.	18	Number(18,4)	No
F_MULTI_UNIT_RTL_AMT	Unit dollar value of new retail multi-unit valuation/price.	18	Number(18,4)	No
F_MULTI_UNIT_RTL_AMT_LCL	Unit value of new retail multiunit valuation/price, in local currency.	18	Number(18,4)	No
SELLING_UOM_CDE	Contains the selling unit of measure code for an item's single-unit retail. This value is non-aggregatable.	4	Character	No
MULTI_SELLING_UOM_CDE	Contains the selling unit of measure code for an item's multi-unit retail. This value is non-aggregatable.	4	Character	No

saviddm.txt-file specification

Business rules:

- This text file contains summarized item availability quantities for a supplier, item on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, supp_idnt, and day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
SUPP_IDNT	The unique identifier for a vendor.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_AVAIL_QTY	The quantity of stock available to be ordered from the supplier.	12	Number(12,4)	No

scmialddm.txt-file specification

Business rules:

- This text file contains data pertaining to a supplier's missed shipments by location and day.
- This text file cannot contain duplicate transactions for the same supp_idnt, loc_idnt, day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_MISSED_ASN_COUNT	The total number of ASN (advanced ship notice) shipments that were expected and not received.	16	Number(16,4)	No

scmiolddm.txt-file specification

Business rules:

- This text file contains data pertaining to a supplier's missed purchase orders by location and day.
- This text file cannot contain duplicate transactions for the same supp_idnt, loc_idnt, day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_MISSED_ORDER_COUNT	The total number of purchase order shipments that were expected and not received.	16	Number(16,4)	No

scrqtlldm.txt-file specification

Business rules:

- This text file contains shipment information about quantity of items received.
- This text file cannot contain duplicate transactions for the same item_idnt, supp_idnt, ship_idnt, loc_idnt, day_dt, po_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
SHIP_IDNT	The unique identifier for a shipment.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
PO_IDNT	The unique identifier for a purchase order.	8	Character	Yes
F_ASN_EXPECTED_QTY	The total quantity expected.	12	Number(12,4)	No
F_RECEIVED_QTY	The total quantity received.	12	Number(12,4)	No
F_ORDERED_QTY	The total quantity ordered.	12	Number(12,4)	No
F_ASN_EXPECTED_COUNT	The number of deliveries where the quantity received equaled the quantity expected (only for ASN).	16	Number(16,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_ASN_UNDER_COUNT	The number of deliveries where the quantity received are less than the number expected (only for ASN).	16	Number(16,4)	No
F_ASN_OVER_COUNT	The number of deliveries where the quantity received exceeded than the number expected (only for ASN).	16	Number(16,4)	No
F_MISMATCHED_COUNT	The number of deliveries where quantity was received for an item that was not expected.	16	Number(16,4)	No
F_FULL_PO_COUNT	The number of purchase orders where all expected quantity was received.	16	Number(16,4)	No
F_PART_PO_COUNT	The number of purchase orders where only part of the expected quantity was received.	16	Number(16,4)	No
F_OVER_PO_COUNT	The number of purchase orders where more than the expected quantity was received.	16	Number(16,4)	No
PICKUP_LOC	User-entered location of shipment for client to pick up.	45	Character	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
PICKUP_NBR	User-entered identifier of a shipment.	25	Character	No
PICKUP_DT	User entered date of the pickup.		Date (YYYYMMDD)	No

scrtlldm.txt-file specification

Business rules:

- This text file contains shipment information about timeliness of receipt.
- This text file cannot contain duplicate transactions for the same item_idnt, supp_idnt, ship_idnt, loc_idnt, day_dt, po_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
SHIP_IDNT	The unique identifier for a shipment.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
PO_IDNT	The unique identifier for a purchase order.	8	Character	Yes
F_ON_TIME_COUNT	The number of deliveries where the quantity received equaled the number expected.	16	Number(16,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_EARLY_COUNT	The number of deliveries that arrived before the scheduled time.	16	Number(16,4)	No
F_LATE_COUNT	The number of deliveries that arrived after the scheduled time.	16	Number(16,4)	No
F_UNSCHEDED_COUNT	The number of deliveries that arrived on days other than the scheduled date.	16	Number(16,4)	No
F_DAYS_EARLY_COUNT	The total number of days a shipment arrived before the scheduled date.	16	Number(16,4)	No
F_DAYS_LATE_COUNT	The total number of days a shipment arrived after the scheduled date.	16	Number(16,4)	No

sctiddm.txt-file specification

Business rules:

- This text file contains supplier contract information.
- Because this datamart is a compressed one, only changes to the contract facts should be included in this text file.
- This text file cannot contain duplicate transactions for the same item_idnt, cntrect_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for the item.	25	Character	Yes
CNTRCT_IDNT	The unique identifier for the contract.	6	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_CNTRCT_QTY	The total contracted quantity to be ordered from the supplier.	12	Number(12,4)	No
F_CNTRCT_COST_AMT	The unit purchase cost negotiated for this contract.	18	Number(18,4)	No
F_CNTRCT_ORD_QTY	The total ordered quantity from the contract to date for all locations.	12	Number(12,4)	No
F_CNTRCT_ORD_COST_AMT	The total cost value for the ordered quantity from the contract to date for all locations.	18	Number(18,4)	No
F_CNTRCT_ORD_CNCLLD_QTY	The total cancelled quantities from the contract to date, for all locations and orders.	12	Number(12,4)	No
F_CNTRCT_ORD_CNCLLD_COST_AMT	The total cost value for the cancelled quantities from the contract to date, for all locations and orders.	18	Number(18,4)	No

sfclwdm.txt-file specification

Business rules:

- This text file contains sales forecast information for an item and location combination on a given week.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, and day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_FCST_SLS_QTY	The forecast sales quantity.	12	Number(12,4)	No

sincilddm.txt-file specification

Business rules:

- This text file contains invoice cost information for each item in a shipment.
- This text file cannot contain duplicate transactions for the same item_idnt, po_idnt, invc_idnt, supp_idnt, ship_idnt, day_dt, loc_idnt, invc_line_nbr combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for the item.	25	Character	Yes
PO_IDNT	The unique identifier for the purchase order.	8	Character	Yes
INVC_IDNT	The unique identifier for the invoice.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for the supplier.	10	Character	Yes
SHIP_IDNT	The unique identifier for the shipment.	10	Character	Yes
DAY_DT	The calendar date for the day the invoice was updated.		Date (YYYYMMDD)	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
INVC_LINE_NBR	Differentiates invoice lines where item-purchase order-supplier-day-ship-location is all the same.	18	Number(18,4)	Yes
F_SUPP_INVC_COST_AMT	The invoice cost in primary currency.	18	Number(18,4)	No
F_SUPP_INVC_COST_AMT_LCL	The invoice cost in local currency.	18	Number(18,4)	No
F_SUPP_INVC_QTY	Quantity of the item shown on the invoice.	12	Number(12,4)	No
SUPP_INVC_STATUS_CDE	Status of the invoice line item. Valid values are 'U' for unmatched, 'R' for partially matched and 'M' for matched.	2	Character	No

slsildmdm.txt-file specification

Business rules:

- This text file contains sales and returns for an item, location, day, minute, voucher, and transaction.
- RDW assumes that tran_idnts received from the source system are unique across location-register-employee-minute-day. Two items, sold at the same location, by the same employee in the same minute, but at two different cash registers to two different customers in two different transactions, will result in two separate and distinct tran_idnts; similarly, the same item/loc/day/minute/register but different employees, ringing up two separate transactions will result in two distinct tran_idnts.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
TRAN_IDNT	The unique identifier for a transaction.	30	Character	Yes
VCHR_IDNT	The unique identifier for a voucher. If the item is a gift certificate, then the corresponding item number will represent a VCHR_IDNT. This attribute is not a dimensional attribute but is used to uniquely identify a record.	16	Character	Yes
DAY_DT	The buseinss date for the day the transaction occurred.	8	Date (YYYYMMDD)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MIN_IDNT	The unique identifier for the minute, made up of the hour_idnt followed by a number 1-60 to indicate the minute of that hour.	4	Number(4) (HHMM)	Yes
OVERRIDE_REASN_CODE_IDNT	The unique identifier for a reason code.	6	Character	Yes
OVERRIDE_REASN_TYPE_IDNT	The unique identifier for a reason type.	6	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
HEAD_IDNT	The unique identifier for a promotion.	10	Character	Yes
SCHM_IDNT	The unique identifier for promotion scheme. Identifies the number belonging to a specific mix and match, threshold, multi-unit, or standard type within a promotion. For item-dept promotions, the schm_idnt must always be -2.	6	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CUST_REF	The customer identifier associated with the transaction.	20	Character	Yes
CUST_REF_TYPE	The type of the identifier number used by a customer.	6	Character	Yes
EMPLY_IDNT	The unique identifier for employee.	10	Character	Yes
SLSPRSN_IDNT	The unique identifier for a salesperson.	10	Character	Yes
CSHR_IDNT	The unique identifier for a cashier.	10	Character	Yes
RGSTR_IDNT	The unique identifier for register.	10	Character	Yes
REASN_CODE_IDNT	The unique identifier for reason code.	6	Character	Yes
REASN_TYPE_IDNT	The unique identifier for reason type.	6	Character	Yes
SUB_TRAN_TYPE_IDNT	The unique identifier for the sub-transaction type.	6	Character	Yes
RTL_TYPE_CDE	Code that indicates whether the retail type is regular, promotion, or clearance.	2	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SLS_AMT	The value of the sale in primary currency.	18	Number(18,4)	No
F_SLS_AMT_LCL	The value of the sale in local currency.	18	Number(18,4)	No
F_SLS_QTY	The number of items involved in the sale.	12	Number(12,4)	No
F_SLS_PRFT_AMT	The profit amount realized on the sale in primary currency	18	Number(18,4)	No
F_SLS_PRFT_AMT_LCL	The profit amount realized on the sale in local currency.	18	Number(18,4)	No
F_RTRN_AMT	The value of the return in primary currency.	18	Number(18,4)	No
F_RTRN_AMT_LCL	The value of the return in local currency.	18	Number(18,4)	No
F_RTRN_QTY	The number of items involved in the return.	12	Number(12,4)	No
F_RTRN_PRFT_AMT	The profit amount realized on the return in primary currency.	18	Number(18,4)	No
F_RTRN_PRFT_AMT_LCL	The profit amount realized on the return in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SLS_ENTER_ITEM_COUNT	Number of times the item is manually entered by cashier for sale.	16	Number(16,4)	No
F_SLS_SCAN_ITEM_COUNT	Number of times the item is scanned by cashier for sale.	16	Number(16,4)	No
F_RTRN_ENTER_ITEM_COUNT	Number of times the item is manually entered by cashier for return.	16	Number(16,4)	No
F_RTRN_SCAN_ITEM_COUNT	Number of times the item is scanned by cashier for return.	16	Number(16,4)	No
F_SLS_IS_MKUP_COUNT	Count of the number of in store markup sales transactions.	16	Number(16,4)	No
F_SLS_IS_MKDN_COUNT	Count of the number of in store markdown sales transactions.	16	Number(16,4)	No
F_RTRN_IS_MKUP_COUNT	Count of the number of in store markup return transactions.	16	Number(16,4)	No
F_RTRN_IS_MKDN_COUNT	Count of the number of in store markdown return transactions.	16	Number(16,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SLS_IS_MKUP_AMT	Total in store markup amount in primary currency for sales transactions.	18	Number(18,4)	No
F_SLS_IS_MKUP_AMT_LCL	Total in store markup amount in local currency for sales transactions.	18	Number(18,4)	No
F_RTRN_IS_MKUP_AMT	Total in store markup amount in primary currency for return transactions.	18	Number(18,4)	No
F_RTRN_IS_MKUP_AMT_LCL	Total in store markup amount in local currency for return transactions.	18	Number(18,4)	No
F_SLS_IS_MKDN_AMT	Total in store markdown amount in primary currency for sales transactions.	18	Number(18,4)	No
F_SLS_IS_MKDN_AMT_LCL	Total in store markdown amount in local currency for sales transactions.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_RTRN_IS_MKDN_AMT	Total in store markdown amount in primary currency for return transactions.	18	Number(18,4)	No
F_RTRN_IS_MKDN_AMT_LCL	Total in store markdown amount in local currency for return transactions.	18	Number(18,4)	No
F_SLS_EMPTY_DISC_AMT	Total employee retail discount amount in primary currency for sales transactions.	18	Number(18,4)	No
F_SLS_EMPTY_DISC_AMT_LCL	Total employee retail discount amount in local currency for sales transactions.	18	Number(18,4)	No
F_RTRN_EMPTY_DISC_AMT	Total employee retail discount amount in primary currency for return transactions.	18	Number(18,4)	No
F_RTRN_EMPTY_DISC_AMT_LCL	Total employee retail discount amount in local currency for return transactions.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SLS_VAT_AMT	The value of the sales value added tax in primary currency.	18	Number(18,4)	No
F_SLS_VAT_AMT_LCL	The value of the sales value added tax in local currency.	18	Number(18,4)	No
F_RTRN_VAT_AMT	The value of the return value added tax in primary currency.	18	Number(18,4)	No
F_RTRN_VAT_AMT_LCL	The value of the return value added tax in local currency.	18	Number(18,4)	No

slsmkdnlddm.txt-file specification

Business rules:

- This text file contains sales markdowns information for an item, location, and retail type on a given day.
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, rtl_type_cde, head_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
RTL_TYPE_CDE	Code that indicates whether the retail type is regular, promotion, or clearance.	2	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
HEAD_IDNT	The unique identifier for a promotion.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_MKDN_AMT	The value of the markdown in primary currency.	18	Number(18,4)	No
F_MKDN_AMT_LCL	The value of the markdown in local currency.	18	Number(18,4)	No
F_MKDN_QTY	The quantity of the markdown.	12	Number(12,4)	No
F_MKUP_AMT	The value of the markup in primary currency.	18	Number(18,4)	No
F_MKUP_AMT_LCL	The value of the markup in local currency.	18	Number(18,4)	No
F_MKUP_QTY	The quantity of the markup.	12	Number(12,4)	No

stlblwdm.txt-file specification

Business rules:

- This text file contains stock ledger values for a department, class, subclass, and location on a given week.
- This text file cannot contain duplicate transactions for the same dept_idnt, class_idnt, sbclass_idnt, and loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SBCLASS_IDNT	The unique identifier for a subclass.	4	Character	Yes
CLASS_IDNT	The unique identifier for a class.	4	Character	Yes
DEPT_IDNT	The unique identifier for a department to which this class belongs in the product hierarchy.	4	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
LOC_TYPE_CDE	Code that indicates whether the location is a store or warehouse.	2	Character	Yes
DAY_DT	The calendar date for the end day of a week in which the transaction occurred.		Date (YYYYMMDD)	Yes
F_IVL_BEG_SOH_COST_AMT	Beginning of week stock on hand total cost, in primary currency	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_BEG_SOH_COST_AMT_LCL	Beginning of week stock on hand total cost, in local currency.	18	Number(18,4)	No
F_IVL_BEG_SOH_RTL_AMT	Beginning of week stock on hand total retail, in primary currency.	18	Number(18,4)	No
F_IVL_BEG_SOH_RTL_AMT_LCL	Beginning of week stock on hand total retail, in local currency	18	Number(18,4)	No
F_IVL_SOH_ADJ_COST_AMT	Value at cost of stock on hand adjustments for a subclass/location during a week, in primary currency.	18	Number(18,4)	No
F_IVL_SOH_ADJ_COST_AMT_LCL	Value at cost of stock on hand adjustments for a subclass/location during a week, in local currency.	18	Number(18,4)	No
F_IVL_SOH_ADJ_RTL_AMT	Value at retail of stock on hand adjustments for a subclass/location during a week, in primary currency.	18	Number(18,4)	No
F_IVL_SOH_ADJ_RTL_AMT_LCL	Value at retail of stock on hand adjustments for a subclass/location during a week, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_RCPTS_COST_AMT	Value at cost of inventory received, in primary currency.	18	Number(18,4)	No
F_IVL_RCPTS_COST_AMT_LCL	Value at cost of inventory received, in local currency.	18	Number(18,4)	No
F_IVL_RCPTS_RTL_AMT	Value at retail of inventory received, in primary currency.	18	Number(18,4)	No
F_IVL_RCPTS_RTL_AMT_LCL	Value at retail of inventory received, in local currency.	18	Number(18,4)	No
F_IVL_RTV_COST_AMT	Value at cost of inventory returned to a vendor, in primary currency.	18	Number(18,4)	No
F_IVL_RTV_COST_AMT_LCL	Value at cost of inventory returned to a vendor, in local currency.	18	Number(18,4)	No
F_IVL_RTV_RTL_AMT	Value at retail of inventory returned to a vendor, in primary currency.	18	Number(18,4)	No
F_IVL_RTV_RTL_AMT_LCL	Value at retail of inventory returned to a vendor, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_TRNSFR_IN_COST_AMT	Value at cost of inventory transferred in, in primary currency.	18	Number(18,4)	No
F_IVL_TRNSFR_IN_COST_AMT_LCL	Value at cost of inventory transferred in, in local currency.	18	Number(18,4)	No
F_IVL_TRNSFR_IN_RTL_AMT	Value at retail of inventory transferred in, in primary currency.	18	Number(18,4)	No
F_IVL_TRNSFR_IN_RTL_AMT_LCL	Value at retail of inventory transferred in, in local currency.	18	Number(18,4)	No
F_IVL_TRNSFR_OUT_COST_AMT	Value at cost of inventory transferred out, in primary currency.	18	Number(18,4)	No
F_IVL_TRNSFR_OUT_COST_AMT_LCL	Value at cost of inventory transferred out, in local currency.	18	Number(18,4)	No
F_IVL_TRNSFR_OUT_RTL_AMT	Value at retail of inventory transferred out, in primary currency.	18	Number(18,4)	No
F_IVL_TRNSFR_OUT_RTL_AMT_LCL	Value at retail of inventory transferred out, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_SHRK_COST_AMT	Value at cost of the difference between actual and ending inventory, in primary currency.	18	Number(18,4)	No
F_IVL_SHRK_COST_AMT_LCL	Value at cost of the difference between actual and ending inventory, in local currency.	18	Number(18,4)	No
F_IVL_SHRK_RTL_AMT	Value at retail of the difference between actual and ending inventory, in primary currency.	18	Number(18,4)	No
F_IVL_SHRK_RTL_AMT_LCL	Value at retail of the difference between actual and ending inventory, in local currency.	18	Number(18,4)	No
F_IVL_RTRNS_COST_AMT	Value at cost of inventory returned from sales, in primary currency.	18	Number(18,4)	No
F_IVL_RTRNS_COST_AMT_LCL	Value at cost of inventory returned from sales, in local currency.	18	Number(18,4)	No
F_IVL_RTRNS_RTL_AMT	Value at retail of inventory returned from sales, in primary currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_RTRNS_RTL_AMT_LCL	Value at retail of inventory returned from sales, in local currency.	18	Number(18,4)	No
F_IVL_RECLASS_IN_COST_AMT	Value of inventory reclassified to this location, valued at cost, in primary currency.	18	Number(18,4)	No
F_IVL_RECLASS_IN_COST_AMT_LCL	Value of inventory reclassified to this location, valued at cost, in local currency.	18	Number(18,4)	No
F_IVL_RECLASS_IN_RTL_AMT	Value of inventory reclassified to this location, valued at retail, in primary currency.	18	Number(18,4)	No
F_IVL_RECLASS_IN_RTL_AMT_LCL	Value of inventory reclassified to this location, valued at retail, in local currency.	18	Number(18,4)	No
F_IVL_RECLASS_OUT_COST_AMT	Value of inventory reclassified from this location, valued at cost, in primary currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_RECLASS_OUT_COST_AMT_LCL	Value of inventory reclassified from this location, valued at cost, in local currency.	18	Number(18,4)	No
F_IVL_RECLASS_OUT_RTL_AMT	Value of inventory reclassified from this location, valued at retail, in primary currency.	18	Number(18,4)	No
F_IVL_RECLASS_OUT_RTL_AMT_LCL	Value of inventory reclassified from this location, valued at retail, in local currency.	18	Number(18,4)	No
F_IVL_SLS_COST_AMT	Value at cost of inventory sold, in primary currency.	18	Number(18,4)	No
F_IVL_SLS_COST_AMT_LCL	Value at cost of inventory sold, in local currency.	18	Number(18,4)	No
F_IVL_SLS_RTL_AMT	Value at retail of inventory sold, in primary currency.	18	Number(18,4)	No
F_IVL_SLS_RTL_AMT_LCL	Value at retail of inventory sold, in local currency.	18	Number(18,4)	No
F_IVL_END_SOH_COST_AMT	End of week stock on hand total cost, in primary currency.	18	Number(18,4)	No
F_IVL_END_SOH_COST_AMT_LCL	End of week stock on hand total cost, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_END_SOH_RTL_AMT	End of week stock on hand total retail, in primary currency.	18	Number(18,4)	No
F_IVL_END_SOH_RTL_AMT_LCL	End of week stock on hand total retail, in local currency.	18	Number(18,4)	No
F_IVL_GRS_PRFT_AMT	Total gross profit amount, in primary currency.	18	Number(18,4)	No
F_IVL_GRS_PRFT_AMT_LCL	Total gross profit amount, in local currency.	18	Number(18,4)	No
F_IVL_CUM_MKON_PCT	The cumulative markon percent.	12	Number(12,4)	No
F_IVL_ACTL_STOCK_COST_AMT	Value at cost of actual stock, only after physical inventory, in primary currency.	18	Number(18,4)	No
F_IVL_ACTL_STOCK_COST_AMT_LCL	Value at cost of actual stock, only after physical inventory, in local currency.	18	Number(18,4)	No
F_IVL_ACTL_STOCK_RTL_AMT	Value at retail of actual stock, only after physical inventory, in primary currency.	18	Number(18,4)	No
F_IVL_ACTL_STOCK_RTL_AMT_LCL	Value at retail of actual stock, only after physical inventory, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_MKUP_AMT	Value of upward revisions in price, in primary currency.	18	Number(18,4)	No
F_IVL_MKUP_AMT_LCL	Value of upward revisions in price, in local currency.	18	Number(18,4)	No
F_IVL_MKUP_CNCLLD_AMT	Value of corrections to a upward revisions in price, in primary currency.	18	Number(18,4)	No
F_IVL_MKUP_CNCLLD_AMT_LCL	Value of corrections to the upward revisions in price, in local currency.	18	Number(18,4)	No
F_IVL_MKDN_CNCLLD_AMT	Value of upward revisions in price, used to offset a previously, in primary currency.	18	Number(18,4)	No
F_IVL_MKDN_CNCLLD_AMT_LCL	Value of upward revisions in price, used to offset a previously, in local currency.	18	Number(18,4)	No
F_IVL_PERM_MKDN_AMT	Value of permanent reduction in price, in primary currency.	18	Number(18,4)	No
F_IVL_PERM_MKDN_AMT_LCL	Value of permanent reduction in price, in local currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_PRMTN_MKDN_AMT	Value of promotion reductions of the price, in primary currency.	18	Number(18,4)	No
F_IVL_PRMTN_MKDN_AMT_LCL	Value of promotion reductions of the price, in local currency.	18	Number(18,4)	No
F_IVL_CLRC_MKDN_AMT	Value of clearance reductions of the price, in primary currency.	18	Number(18,4)	No
F_IVL_CLRC_MKDN_AMT_LCL	Value of clearance reductions of the price, in local currency.	18	Number(18,4)	No
F_IVL_EMPTY_DISC_AMT	Value of employee discounts, in primary currency.	18	Number(18,4)	No
F_IVL_EMPTY_DISC_AMT_LCL	Value of employee discounts, in local currency.	18	Number(18,4)	No
F_IVL_CASH_DISC_AMT	Value of cash discounts, in primary currency.	18	Number(18,4)	No
F_IVL_CASH_DISC_AMT_LCL	Value of cash discounts, in local currency.	18	Number(18,4)	No
F_IVL_FRGHT_COST_AMT	Value of freight expenses, in primary currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_IVL_FRGHT_COST_ AMT_LCL	Value of freight expenses, in local currency.	18	Number(18,4)	No
F_IVL_WRKRM_COST_ AMT	Value of workroom expenses, in primary currency.	18	Number(18,4)	No
F_IVL_WRKRM_COST_ AMT_LCL	Value of workroom expenses, in local currency.	18	Number(18,4)	No
F_IVL_GAFS_COST_ AMT	Goods available for sale valued at cost, in primary currency.	18	Number(18,4)	No
F_IVL_GAFS_COST_ AMT_LCL	Goods available for sale valued at cost, in local currency.	18	Number(18,4)	No
F_IVL_GAFS_RTL_AMT	Goods available for sale valued at retail, in primary currency.	18	Number(18,4)	No
F_IVL_GAFS_RTL_ AMT_LCL	Goods available for sale valued at retail, in local currency.	18	Number(18,4)	No

tldmdm.txt-file specification

Business rules:

- This text file contains tender type transaction information.
- Amounts will be summed in the target table by tndr_type_idnt, tran_idnt, loc_idnt, day_dt, min_idnt, rgstr_idnt, and cshr_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type.	6	Character	Yes
TRAN_IDNT	The unique identifier for the transaction.	30	Character	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
MIN_IDNT	The unique identifier for the minute of the time the transaction occurred. This field is made up of the hour_idnt followed by a number 1-60 to indicate the minute of that hour.	4	Number(4)	Yes
RGSTR_IDNT	The unique identifier for the register.	10	Character	Yes
CSHR_IDNT	The unique identifier for the cashier.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_CC_SCAN_FLAG	Indicates whether the credit card was scanned or manually entered. Valid values are 'Y' for scanned, or 'N' or Null for manually entered.	1	Character	No
F_TNDR_COUPON_COUNT	Total count of tender coupons used per transaction. Tender coupons are issued by the manufacturer as opposed to the store.	16	Number(16,4)	No
F_TNDR_COUPON_AMT	Total amount of tender coupons used per transaction in primary currency. Tender coupons are issued by the manufacturer as opposed to the store.	18	Number(18,4)	No
F_TNDR_COUPON_AMT_LCL	Total amount of tender coupons used per transaction in local currency. Tender coupons are issued by the manufacturer as opposed to the store.	18	Number(18,4)	No
F_TNDR_SLS_AMT	Sales amount paid for with a particular tender type in primary currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_TNDR_SLS_AMT_LCL	Sales amount paid for with a particular tender type in local currency.	18	Number(18,4)	No
F_TNDR_RTRNS_SLS_AMT	Return amount credited to a particular tender type in primary currency.	18	Number(18,4)	No
F_TNDR_RTRNS_SLS_AMT_LCL	Return amount credited to a particular tender type in local currency.	18	Number(18,4)	No

vchreschddm.txt-file specification

Business rules:

- This text file contains the date and count of escheat vouchers. When a voucher escheats, the retailer releases all liability of the voucher to the state government. The quantity of escheated vouchers and the dates on which they escheated are captured from this text file.
- This text file cannot contain duplicate transactions for the same day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_ESCH_COUNT	The total count of escheated vouchers on a particular day.	16	Number(16,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_ESCH_AMT	The monetary amount of the escheated vouchers. If the voucher was never issued, the escheat amount is 0. If it was issued, the escheat amount is the issue amount.	18	Number(18,4)	No

vchrmoveldsg.txt-file specification

Business rules:

- This text file contains issued and redeemed voucher information at the individual voucher level.
- This text file cannot contain duplicate transactions for the same vchr_line_no, vchr_status_cde combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
VCHR_LINE_NO	The unique identifier for the voucher.	20	Character	Yes
VCHR_STATUS_CDE	Indicates whether this is an issue (I) or redemption (R) record for this voucher.	1	Character	Yes
LOC_IDNT	The unique identifier for the location.	10	Character	Yes (–1 if no location)
DAY_DT	The calendar date for the day the status of the voucher was captured.		Date (YYYYMMDD)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
VCHR_AGE	The age of the voucher in days.	6	Number(6)	Yes
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type. An example of a tender type is Discover Card, Master Card, or Visa.	6	Character	Yes
RGSTR_IDNT	The unique identifier for the register.	10	Character	Yes (–1 if no register)
CSHR_IDNT	The unique identifier for the cashier.	10	Character	Yes (–1 if no cashier)
F_AMT	The monetary amount for which this voucher was issued/redeemed in primary currency.	18	Number(18,4)	No
F_AMT_LCL	The monetary amount for which this voucher was issued/redeemed in the issue/redemption location's local	18	Number(18,4)	No

vchroutlwdm.txt-file specification

Business rules:

- This text file contains outstanding voucher information ‘as of’ the day_dt. A voucher is outstanding if it has been issued but not yet redeemed or escheated (that is, fully outstanding).
- This text file cannot contain duplicate transactions for the same loc_idnt, week, vchr_age, tndr_type_id_idnt, rgstr_idnt, cshr_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
DAY_DT	The calendar date for the day the voucher was issued.		Date (YYYYMMDD)	Yes
VCHR_AGE	The age of the voucher in days.	6	Number(6)	Yes
TNDR_TYPE_ID_IDNT	The unique identifier for the tender type. An example of a tender type is Discover Card, Master Card, or Visa.	6	Character	Yes
RGSTR_IDNT	The unique identifier for the register.	10	Character	Yes
CSHR_IDNT	The unique identifier for the cashier.	10	Character	Yes
F_OUT_COUNT	The number of outstanding vouchers.	16	Number(16,4)	No
F_OUT_AMT	The monetary amount of the outstanding vouchers in primary currency.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_OUT_AMT_LCL	The monetary amount of the outstanding vouchers in local currency.	18	Number(18,4)	No

Load only

mslsdlwdm.txt-file specification

Business rules:

- This text file contains market sales data for a market category and market area level on a given week.
- This text file cannot contain duplicate transactions for the same mkt_dept_idnt, mkt_area_level_idnt, and wk_end_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_DEPT_IDNT	The unique identifier for a market department.	13	Character	Yes
MKT_AREA_LEVEL_IDNT	The unique identifier for a market area level.	16	Character	Yes
WK_END_DT	The date in which the week ends.		Date (YYYYMMDD)	Yes
MKT_GEO_LEVEL	The market geographic level. Valid values are 1 or 2 or 3.	1	Character	Yes
MKT_RECD_CURR_DT	The market data creation date.		Date (YYYYMMDD)	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_SLS_AMT_LCL	Total sales of the market item in local currency for the week.	18	Number(18,4)	No
F_MKT_SLS_AMT	Total sales of the market item primary currency for the week.	18	Number(18,4)	No
F_MKT_SLS_QTY	Total number of the market item sold for the week.	12	Number(12,4)	No
F_MKT_AVG_ACV_WGT_DIST_PCT	Average weekly all commodity volume weighted distribution. A measure of the percent of stores stocking the product, weighted by all commodity volume.	12	Number(12,4)	No
F_MKT_AVG_MMACHV_SLS_RATE	Average weekly sales value per \$MM (million dollar) all commodity volume (sales rate). The sales efficiency of the product in relation to its distribution, based on all commodity volume per \$MM.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_AVG_WGT_PRICE_REDT_PCT	Weighted average percent price reduction. The average amount the retail was reduced for stores selling the item, weighted by units sold at each retail.	12	Number(12,4)	No
F_MKT_AVG_STORE_SELL_ITEM_QTY	Average weekly items per stores selling. The average number of different UPCs of a selected product available in each store.	12	Number(12,4)	No
F_MKT_NORMAL_AMT_LCL	Estimated sales in local currency that would have been recorded if there were no impact from display, promotion or price reduction for the week.	18	Number(18,4)	No
F_MKT_NORMAL_AMT	Estimated sales in primary currency that would have been recorded if there were no impact from display, promotion or price reduction for the week.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_NORMAL_QTY	Estimated sales units that would have been recorded if there were no impact from display, promotion or price reduction for the week.	12	Number(12,4)	No
F_MKT_SLS_PRICE_CUT_AMT_LCL	Sales main ad or price cut in local currency. The total sales value for any item on feature, display and/or with price reductions	18	Number(18,4)	No
F_MKT_SLS_PRICE_CUT_AMT	Sales main ad or price cut in primary currency. The total sales value for any item on feature, display and/or with price reductions.	18	Number(18,4)	No
F_MKT_SLS_PRICE_CUT_QTY	Unit sales main ad or price cut. The total unit sales for any item on feature, display and/or with price reductions.	12	Number(12,4)	No
F_MKT_MAIN_AD_AMT_LCL	The total sales in local currency for any item on feature.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_MAIN_AD_AMT	The total sales in primary currency for any item on feature.	18	Number(18,4)	No
F_MKT_MAIN_AD_QTY	The total unit sales for any item on feature.	12	Number(12,4)	No

mslsilwdm.txt-file specification

Business rules:

- This text file contains market sales data at the market item, market area level, and week level.
- This text file cannot contain duplicate transactions for the same mkt_item_idnt, mkt_area_level_idnt, wk_end_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
MKT_ITEM_IDNT	The unique identifier for a market item.	25	Character	Yes
MKT_AREA_LEVEL_IDNT	The unique identifier for market area level.	16	Character	Yes
WK_END_DT	The date on which the week ends.		Date (YYYYMMDD)	Yes
MKT_GEO_LEVEL	The market geographic level.	1	Character	Yes
MKT_RECD_CURR_DT	The date this record is current in the source system.		Date (YYYYMMDD)	Yes
F_MKT_SLS_AMT_LCL	Total sales of the market item in local currency for the week.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_SLS_AMT	Total sales of the market item primary currency for the week.	18	Number(18,4)	No
F_MKT_SLS_QTY	Total number of market items sold.	12	Number(12,4)	No
F_MKT_AVG_ACV_WGT_DIST_PCT	Average weekly all commodity volume weighted distribution. A measure of the percent of stores stocking the product, weighted by all commodity volume.	12	Number(12,4)	No
F_MKT_AVG_MMACHV_SLS_RATE	Average weekly sales value per \$MM (million dollar) all commodity volume (sales rate). The sales efficiency of the product in relation to its distribution, based on all commodity volume per \$MM.	12	Number(12,4)	No
F_MKT_AVG_WGT_PRICE_REDT_PCT	Weighted average percent price reduction. The average amount the retail was reduced for stores selling the item, weighted by units sold at each retail.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_AVG_STORE_SELL_ITEM_QTY	Average weekly items per store selling. The average number of different UPCs of a selected product available in each store carrying the product.	12	Number(12,4)	No
F_MKT_NORMAL_AMT_LCL	Estimated sales in local currency that would have been recorded if there were no impact from display, promotion or price reduction for the week.	18	Number(18,4)	No
F_MKT_NORMAL_AMT	Estimated sales in primary currency that would have been recorded if there were no impact from display, promotion or price reduction for the week.	18	Number(18,4)	No
F_MKT_NORMAL_QTY	Estimated sales units that would have been recorded if there were no impact from display, promotion or price reduction.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_MKT_SLS_PRICE_CUT_AMT_LCL	Sales main ad or price cut in local currency. The total sales value for any item on feature, display and/or with price reductions.	18	Number(18,4)	No
F_MKT_SLS_PRICE_CUT_AMT	Sales main ad or price cut in primary currency. The total sales value for any item on feature, display and/or with price reductions.	18	Number(18,4)	No
F_MKT_SLS_PRICE_CUT_QTY	Unit sales main ad or price cut. The total unit sales for any item on feature, display and/or with price reductions.	12	Number(12,4)	No
F_MKT_MAIN_AD_AMT_LCL	The total sales in local currency for any item on feature.	18	Number(18,4)	No
F_MKT_MAIN_AD_AMT	The total sales in primary currency for any item on feature.	18	Number(18,4)	No
F_MKT_MAIN_AD_QTY	The total unit sales for any item on feature.	12	Number(12,4)	No

plcblwdm.txt-file specification

Business rules:

- This text file contains current planning data for a department, class, subclass and location on a given week.
- This text file cannot contain duplicate transactions for the same day_dt, dept_idnt, class_idnt, subclass_idnt, and loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the end day of a week in which the transaction occurred.		Date (YYYYMMDD)	Yes
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes
CLASS_IDNT	The unique identifier for a class to which this item belongs in the product hierarchy.	4	Character	No
SBCLASS_IDNT	The unique identifier for a subclass to which this item belongs in the product hierarchy.	4	Character	No
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
F_PLN_CURR_SLS_QTY	Current plan total sales quantity, which includes regular, clearance and promotional sales minus customer returns.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_SLS_AMT	Current plan total sales amount, which includes regular, clearance and promotional sales minus customer returns.	18	Number(18,4)	No
F_PLN_CURR_GRS_PRFT_AMT	Current plan gross margin amount.	18	Number(18,4)	No
F_PLN_CURR_RGLR_MKDN_AMT	Current plan regular markdown amount.	18	Number(18,4)	No
F_PLN_CURR_CLRC_MKDN_AMT	Current plan clearance markdown amount.	18	Number(18,4)	No
F_PLN_CURR_PRMTN_MKDN_AMT	Current plan promotion markdown amount.	18	Number(18,4)	No
F_PLN_CURR_SHRK_QTY	Current plan shrinkage units, the total units of loss of inventory over time due to damage, misplacement, or theft.	12	Number(12,4)	No
F_PLN_CURR_SHRK_RTL_AMT	Current plan shrinkage retail value, the total retail value of loss of inventory over time due to damage, misplacement, or theft.	18	Number(18,4)	No
F_PLN_CURR_BOP_QTY	Current plan beginning inventory units.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_BOP_COST_AMT	Current plan beginning inventory cost amount.	18	Number(18,4)	No
F_PLN_CURR_BOP_RTL_AMT	Current plan beginning inventory retail amount.	18	Number(18,4)	No
F_PLN_CURR_OTB_QTY	Current plan quantity of goods that may be received in stock without exceeding planned inventory levels.	12	Number(12,4)	No
F_PLN_CURR_OTB_COST_AMT	Current plan cost of goods that may be received in stock without exceeding planned inventory levels.	18	Number(18,4)	No
F_PLN_CURR_OTB_RTL_AMT	Current plan retail of goods that may be received in stock without exceeding planned inventory levels.	18	Number(18,4)	No
F_PLN_CURR_RCPTS_QTY	Current plan quantity of goods to be received in stock.	12	Number(12,4)	No
F_PLN_CURR_RCPTS_COST_AMT	Current plan cost of planned quantity of goods to be received in stock.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_RCPTS_RTL_AMT	Current plan retail of planned quantity of goods to be received in stock.	18	Number(18,4)	No
F_PLN_CURR_CMTS_RTL_AMT	Current plan retail amount of commitments made to suppliers.	18	Number(18,4)	No
F_PLN_CURR_ORD_CNCLLD_RTL_AMT	Current plan on order cancel retail amount.	18	Number(18,4)	No
F_PLN_CURR_ORD_RTL_AMT	Current plan retail of goods that have been ordered but not received.	18	Number(18,4)	No
F_PLN_CURR_RECL_IN_RTL_AMT	Current plan retail amount of inventory transferred in as a result of reclassification.	18	Number(18,4)	No
F_PLN_CURR_RECL_OUT_RTL_AMT	Current plan retail amount of inventory transferred out as a result of reclassification.	18	Number(18,4)	No
F_PLN_CURR_RCVD_RTL_AMT	Current plan retail of goods received into inventory.	18	Number(18,4)	No
F_PLN_CURR_RTV_RTL_AMT	Current plan goods returned to vendor expressed in retail amount.	18	Number(18,4)	No
F_PLN_CURR_CMTS_QTY	Current plan units ordered but not approved.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_ORD_ CNCLLD_QTY	Current plan cancelled orders expressed in units.	12	Number(12,4)	No
F_PLN_CURR_ORD_ QTY	Current plan quantity of goods that have been ordered but not received.	12	Number(12,4)	No
F_PLN_CURR_RECL_ IN_QTY	Current plan quantity of inventory transferred in as a result of reclassification.	12	Number(12,4)	No
F_PLN_CURR_RECL_ OUT_QTY	Current plan quantity of inventory transferred out as a result of reclassification.	12	Number(12,4)	No
F_PLN_CURR_RCVD_ QTY	Current plan goods received into inventory.	12	Number(12,4)	No
F_PLN_CURR_RTV_ QTY	Current plan goods returned to vendor expressed in units.	12	Number(12,4)	No
F_PLN_CURR_EOP_ RTL_AMT	Current plan ending inventory retail amount.	18	Number(18,4)	No
F_PLN_CURR_WOS_ AMT	Current plan weeks of supply: ratio of beginning inventory value to sales value on a weekly basis.	18	Number(18,4)	No
F_PLN_CURR_EOP_ COST_AMT	Current plan ending inventory cost amount.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_ORD_CNCLLD_COST_AMT	Current plan on order cancel cost amount.	18	Number(18,4)	No
F_PLN_CURR_ORD_COST_AMT	Current plan cost of goods that have been ordered but not received.	18	Number(18,4)	No
F_PLN_CURR_RCVD_COST_AMT	Current plan cost of goods received into inventory.	18	Number(18,4)	No
F_PLN_CURR_CMTS_COST_AMT	Current plan cost amount of commitments made to suppliers.	18	Number(18,4)	No
F_PLN_CURR_CUM_MKUP_AMT	Current plan percentage difference between total delivered cost and total original retail value of merchandise handled within a stated time frame, inclusive of the accumulated inventory.	18	Number(18,4)	No
F_PLN_CURR_EOP_QTY	Current plan ending inventory units.	12	Number(12,4)	No
F_PLN_CURR_WOS_QTY	Current plan weeks of supply: ratio of beginning inventory units to sales units on a weekly basis.	12	Number(12,4)	No
F_PLN_CURR_COGS_AMT	Current plan cost of goods sold amount.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_CURR_SLS_VAT_AMT	Current plan total value added tax amount, which includes regular, clearance and promotional sales minus customer returns.	18	Number(18,4)	No
F_PLN_CURR_EMPLY_DISC_AMT	Current plan employee discount at retail.	18	Number(18,4)	No
F_PLN_CURR_FRGHT_COST_AMT	Current plan freight cost amount.	18	Number(18,4)	No
F_PLN_CURR_WRKRM_COST_AMT	Current plan workroom cost amount.	18	Number(18,4)	No
F_PLN_CURR_RTRNS_SLS_AMT	Current plan customer sales return retail amount.	18	Number(18,4)	No

ploblwdm.txt-file specification

Business rules:

- This text file contains original planning data for a department, class, subclass, and location on a given week.
- This text file cannot contain duplicate transactions for the same day_dt, dept_idnt, class_idnt, sbclass_idnt, and loc_idnt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the end day of a week in which the transaction occurred.		Date (YYYYMMDD)	Yes
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
CLASS_IDNT	The unique identifier for a class to which this item belongs in the product hierarchy.	4	Character	No
SBCLASS_IDNT	The unique identifier for a subclass to which this item belongs in the product hierarchy.	4	Character	No
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
F_PLN_ORIG_SLS_QTY	Original plan total sales quantity, which includes regular, clearance and promotional sales minus customer returns.	12	Number(12,4)	No
F_PLN_ORIG_SLS_AMT	Original plan total sales amount, which includes regular, clearance and promotional sales minus customer returns.	18	Number(18,4)	No
F_PLN_ORIG_GRS_PRFT_AMT	Original plan gross margin amount.	18	Number(18,4)	No
F_PLN_ORIG_RGLR_MKDN_AMT	Original plan regular markdown amount.	18	Number(18,4)	No
F_PLN_ORIG_CLRC_MKDN_AMT	Original plan clearance markdown amount.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_ORIG_PRMTN_MKDN_AMT	Original plan promotion markdown amount.	18	Number(18,4)	No
F_PLN_ORIG_SHRK_QTY	Original plan shrinkage units, the total units of loss of inventory over time due to damage, misplacement, or theft.	12	Number(12,4)	No
F_PLN_ORIG_SHRK_RTL_AMT	Original plan shrinkage retail value, the total retail value of loss of inventory over time due to damage, misplacement, or theft.	18	Number(18,4)	No
F_PLN_ORIG_BOP_QTY	Original plan beginning inventory units.	12	Number(12,4)	No
F_PLN_ORIG_BOP_COST_AMT	Original plan beginning inventory cost amount.	18	Number(18,4)	No
F_PLN_ORIG_BOP_RTL_AMT	Original plan beginning inventory retail amount.	18	Number(18,4)	No
F_PLN_ORIG_RCPTS_QTY	Original plan quantity of goods that may be received in stock without exceeding planned inventory levels.	12	Number(12,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_ORIG_RCPTS_COST_AMT	Original plan cost of goods that may be received in stock without exceeding planned inventory levels.	18	Number(18,4)	No
F_PLN_ORIG_RCPTS_RTL_AMT	Original plan retail of goods that may be received in stock without exceeding planned inventory levels.	18	Number(18,4)	No
F_PLN_ORIG_CMTS_RTL_AMT	Original plan retail amount of commitments made to suppliers.	18	Number(18,4)	No
F_PLN_ORIG_ORD_CNCLD_COST_AMT	Original plan on order cancel cost amount.	18	Number(18,4)	No
F_PLN_ORIG_ORD_RTL_AMT	Original plan retail of goods that have been ordered but not received.	18	Number(18,4)	No
F_PLN_ORIG_RECL_IN_RTL_AMT	Original plan retail amount of inventory transferred in as a result of reclassification.	18	Number(18,4)	No
F_PLN_ORIG_RECL_OUT_RTL_AMT	Original plan retail amount of inventory transferred out as a result of reclassification.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_ORIG_RCVD_RTL_AMT	Original plan retail of goods received into inventory.	18	Number(18,4)	No
F_PLN_ORIG_RTV_RTL_AMT	Original plan goods returned to vendor expressed in retail amount.	18	Number(18,4)	No
F_PLN_ORIG_CMTS_QTY	Original plan units ordered but not approved.	12	Number(12,4)	No
F_PLN_ORIG_ORD_CNCLLD_QTY	Original plan cancelled orders expressed in units.	12	Number(12,4)	No
F_PLN_ORIG_ORD_QTY	Original plan quantity of goods that have been ordered but not received.	12	Number(12,4)	No
F_PLN_ORIG_RECL_IN_QTY	Original plan quantity of inventory transferred in as a result of reclassification.	12	Number(12,4)	No
F_PLN_ORIG_RECL_OUT_QTY	Original plan quantity of inventory transferred out as a result of reclassification.	12	Number(12,4)	No
F_PLN_ORIG_RCVD_QTY	Original plan goods received into inventory.	12	Number(12,4)	No
F_PLN_ORIG_RTV_QTY	Original plan goods returned to vendor expressed in units.	12	Number(12,4)	No
F_PLN_ORIG_EOP_RTL_AMT	Original plan ending inventory retail amount.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_ORIG_EOP_QTY	Original plan ending inventory units.	12	Number(12,4)	No
F_PLN_ORIG_ORD_COST_AMT	Original plan cost of goods that have been ordered but not received.	18	Number(18,4)	No
F_PLN_ORIG_ORD_CNCLD_RTL_AMT	Original plan on order cancel retail amount.	18	Number(18,4)	No
F_PLN_ORIG_CMTS_COST_AMT	Original plan cost amount of commitments made to suppliers.	18	Number(18,4)	No
F_PLN_ORIG_RCVD_COST_AMT	Original plan cost of goods received into inventory.	18	Number(18,4)	No
F_PLN_ORIG_CUM_MKUP_AMT	Original plan percentage difference between total delivered cost and total original retail value of merchandise handled within a stated time frame, inclusive of the accumulated inventory.	18	Number(18,4)	No
F_PLN_ORIG_COGS_AMT	Original plan cost of goods sold amount.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_PLN_ORIG_SLS_VAT_AMT	Original plan total value added tax amount, which includes regular, clearance and promotional sales minus customer returns.	18	Number(18,4)	No
F_PLN_ORIG_EMPTY_DISC_AMT	Original plan employee discount at retail.	18	Number(18,4)	No
F_PLN_ORIG_FRGHT_COST_AMT	Original plan freight cost amount.	18	Number(18,4)	No
F_PLN_ORIG_WKRM_COST_AMT	Original plan workroom cost amount.	18	Number(18,4)	No
F_PLN_ORIG_RTRNS_SLS_AMT	Original plan customer sales return retail amount.	18	Number(18,4)	No
F_PLN_ORIG_EOP_COST_AMT	Original plan ending cost amount.	18	Number(18,4)	No

scmidddm.txt-file specification

Business rules:

- This text file contains data pertaining to a supplier's missed deliveries by location and day.
- This text file cannot contain duplicate transactions for the same supp_idnt, loc_idnt, day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_MISSED_SCHED_COUNT	The total number of scheduled shipments that have not been received.	16	Number(16,4)	No

scqcdm.txt-file specification

Business rules:

- This text file contains shipment information about which items requiring QC (quality control) failed or passed the QC test.
- This text file cannot contain duplicate transactions for the same item_idnt, ship_idnt, supp_idnt, loc_idnt, day_dt, po_idnt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
SHIP_IDNT	The unique identifier for a shipment.	10	Character	Yes
SUPP_IDNT	The unique identifier for a supplier.	10	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
PO_IDNT	The unique identifier for a purchase order.	8	Character	Yes

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_QC_FLAG	Indicates whether or not quality control checking was required on the receipt.	1	Character	No
F_QC_FAILED_QTY	The total quantity of items that failed quality control checks.	12	Number(12,4)	No
F_QC_PASSED_QTY	The total quantity of items that passed quality control checks.	12	Number(12,4)	No

spaldlddm.txt-file specification

Business rules:

- This text file contains information about the amount of space allocated for each department at a particular location on a particular day. The space is measured in one, two, or three-dimensional space (linear, square, cubic).
- This text file cannot contain duplicate transactions for the same dept_idnt, loc_idnt, day_dt.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
DEPT_IDNT	The unique identifier for a department.	4	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_SA_LINEAR_AMT	The amount of linear space.	18	Number(18,4)	No
F_SA_SQUARE_AMT	The amount of square space.	18	Number(18,4)	No
F_SA_CUBIC_AMT	The amount of cubic space.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SA_LINEAR_MAX_AMT	The maximum amount of linear space.	18	Number(18,4)	No
F_SA_SQUARE_MAX_AMT	The maximum amount of square space.	18	Number(18,4)	No
F_SA_CUBIC_MAX_AMT	The maximum amount of cubic space.	18	Number(18,4)	No
F_SA_LINEAR_MIN_AMT	The minimum amount of linear space.	18	Number(18,4)	No
F_SA_SQUARE_MIN_AMT	The minimum amount of square space.	18	Number(18,4)	No
F_SA_CUBIC_MIN_AMT	The minimum amount of cubic space.	18	Number(18,4)	No
F_SA_FACINGS	The number of facings for a display.	18	Number(18,4)	No
F_SA_ON_DISP_IND	Indicates whether an item is on display or not.	1	Character	No
F_SA_ON_FEAT_IND	Indicates whether an item is on feature or not.	1	Character	No

spaliiddm.txt-file specification

Business rules:

- This text file contains information about the amount of space allocated for each item at a particular location on a particular day. The space is measured in one, two or three-dimensional space (linear, square, cubic).
- This text file cannot contain duplicate transactions for the same item_idnt, loc_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
ITEM_IDNT	The unique identifier for an item.	25	Character	Yes
LOC_IDNT	The unique identifier for a location.	10	Character	Yes
DAY_DT	The calendar date for the day the transaction occurred.		Date (YYYYMMDD)	Yes
F_SA_LINEAR_AMT	The amount of linear space allotted to the item at the location, expressed in the customer's preferred unit of measure.	18	Number(18,4)	No
F_SA_SQUARE_AMT	The amount of two-dimensional space allotted to the item (such as square feet or square centimeters) at the location, expressed in the customer's preferred unit of measure).	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SA_CUBIC_AMT	The amount of three-dimensional space allotted to the item (such as cubic feet or cubic centimeters) at the location, expressed in the customer's preferred unit of measure.	18	Number(18,4)	No
F_SA_LINEAR_MAX_AMT	The max amount of linear space allotted to the item at the location, expressed in the customer's preferred unit of measure.	18	Number(18,4)	No
F_SA_SQUARE_MAX_AMT	The max amount of two-dimensional space allotted to the item (such as square feet or square centimeters) at the location, expressed in the customer's preferred unit of measure.	18	Number(18,4)	No

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
F_SA_CUBIC_MAX_AMT	The max amount of three-dimensional space allotted to the item (such as cubic feet or cubic centimeters) at the location, expressed in the customer's preferred unit of measure.	18	Number(18,4)	No
F_SA_LINEAR_MIN_AMT	The minimum amount of linear space.	18	Number(18,4)	No
F_SA_SQUARE_MIN_AMT	The minimum amount of square space.	18	Number(18,4)	No
F_SA_CUBIC_MIN_AMT	The minimum amount of cubic space.	18	Number(18,4)	No
F_SA_FACINGS	The number of facings for a display.	18	Number(18,4)	No
F_SA_ON_DISP_IND	Indicates whether an item is on display or not.	1	Character	No
F_SA_ON_FEAT_IND	Indicates whether an item is on feature or not.	1	Character	No

sttflddm.txt-file specification

Business rules:

- This text file contains store traffic information.
- This text file cannot contain duplicate transactions for the same loc_idnt, day_dt combination.

FIELD NAME	DESCRIPTION	MAX COLUMN LENGTH	DATA TYPE / FORMAT	REQUIRED FIELD
LOC_IDNT	The unique identifier for the location.	10	Character	Yes
DAY_DT	The calendar date for the day the store was visited.		Date (YYYYMMDD)	Yes
F_STORE_TRAFFIC	The number of visitors to a particular store on a certain day.	16	Number(16,4)	No