Oracle® Retail Pricing Cloud Service

Operations Guide

Release 16.0.030

F19524-02

March 2020

ORACLE®

Oracle Retail Price Management Operations Guide, Release 16.0.030

F19524-02

Primary Author: Nathan Young

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

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

Contents

Send Us Your Comments ix

Preface xi

Audience xi
Documentation Accessibility xi
Related Documents xi
Customer Support xi
Review Patch Documentation xii
Improved Process for Oracle Retail Documentation Corrections xii
Oracle Retail Documentation on the Oracle Technology Network xii
Conventions xii

1 Introduction

2 Technical Architecture

Overview 2-1
Oracle Application Development Framework (ADF) 2-1
Model-View-Controller (MVC) Architectural Pattern 2-1
ADF Security 2-2
ADF View (ADFv) 2-2
ADF Controller (ADFc) 2-2
ADF Business Components (ADFbc) 2-2
ADF Model (ADFm) 2-3
Oracle Metadata Services (MDS) 2-3
Retail Fusion Platform 2-3
RPCS Backend Service 2-4
Data Access Patterns 2-4
Database Access Using ADFbc 2-4
Connection Pooling 2-4
Data Storage 2-4
Accessing Merchandising System Data in Real Time 2-4

3 Pricing Batch Processes

BDI Clearance Publishing (BDI_PRICING_CLR_TX _JOB) 3-2
Scheduling Constraints 3-2

Restart/Recovery 3-2
Key Tables Affected 3-2
Design Assumptions 3-2
Output 3-3
BDI Price Change Publishing (BDI_PRICING_PC_TX_CYCLE_JOB) 3-3
Scheduling Constraints 3-3
Restart/Recovery 3-4
Key Tables Affected 3-4
Design Assumptions 3-4
Output 3-4
Promotion Publishing (BDI_PRICING_PROMO_OFFER_TX_CYCLE_JOB) 3-5
Scheduling Constraints 3-5
Restart/Recovery 3-5
Promotions Integration 3-5
Payload Tables 3-6
Payload Population Logic 3-7
BDI Tables 3-9
Key Tables Affected 3-9
Output 3-10
PRC_PAYLD_MSG_HDR_OUT 3-10
PROMO_OFFER_OUT 3-10
PROMO_OFFER_COND_OUT 3-11
PROMO_OFR_REWARD_OUT 3-11
PROM_OFR_CND_MRCH_OUT 3-12
PROMO_OFFER_LOC_OUT 3-13
PROMO_OFR_CANCEL_OUT 3-13
PROM_OFR_CNCL_ITM_OUT 3-13
PROM_OFR_CNCL_LOC_OUT 3-14
ClearanceInductionBatch (Clearance Induction Batch) 3-15
Design Overview 3-15
Scheduling Constraints 3-16
Restart/Recovery 3-16
Key Tables Affected 3-16
Design Assumptions 3-16
ClearancePriceChangePublishBatch (Clearance Price Change Publish Batch) 3-16
Design Overview 3-17
Scheduling Constraints 3-17
Restart/Recovery 3-17
Key Tables Affected 3-17
Output File 3-17
Output File Layout 3-18
Design Assumptions 3-19
FutureRetailPurgeBatch Design 3-19
Design Overview 3-19
Scheduling Constraints 3-19
Restart/Recovery 3-19
Key Tables Affected 3-19

Design Assumptions 3-20
FutureRetailRollUpBatch (Future Retail Roll Up Batch) 3-20
Design Overview 3-20
Scheduling Constraints 3-20
Restart/Recovery 3-20
Key Tables Affected 3-20
Design Assumptions 3-21
ItemReclassBatch (Item Reclass Batch) 3-21
Design Overview 3-21
Scheduling Constraints 3-21
Restart/Recovery 3-21
Key Tables Affected 3-21
Design Assumptions 3-22
NewItemLocationBatch (New Item Location Batch Batch) 3-22
Design Overview 3-22
Scheduling Constraints 3-22
Restart/Recovery 3-23
Key Tables Affected 3-23
Design Assumptions 3-24
NightlyBatchCleanup (Nightly Cleanup Batch) 3-24
Design Overview 3-24
Scheduling Constraints 3-24
Restart/Recovery 3-24
Key Tables Affected 3-25
Design Assumptions 3-25
PriceChangeInductionBatch (Price Change Induction Batch) 3-25
Design Overview 3-26
Scheduling Constraints 3-26
Restart/Recovery 3-26
Key Tables Affected 3-26
Design Assumptions 3-27
PriceEventExecutionBatch (Price Event Execution Batch) 3-27
Design Overview 3-27
Scheduling Constraints 3-27
Restart/Recovery 3-28
Key Tables Affected 3-28
Design Assumptions 3-28
Purge Batch (PurgeBatch) 3-28
System Options 3-29
Usage 3-29
Scheduling Constraints 3-29
Restart/Recovery 3-29
Key Tables Affected 3-29
PurgeGTTCaptureBatch (Purge GTT Capture Batch) 3-32
Design Overview 3-32
Scheduling Constraints 3-32
Restart/Recovery 3-32

Key Tables Affected 3-33
Design Assumptions 3-33
RegularPriceChangePublishBatch (Regular Price Change Publish Batch) 3-33
Design Overview 3-33
Scheduling Constraints 3-34
Restart/Recovery 3-34
Key Tables Affected 3-34
Output Files 3-34
Output File Layout 3-35
Design Assumptions 3-36

4 Induction Services

Price Change Induction Service 4-1
Service Type 4-1
REST URL 4-1
Input Parameters 4-1
JSON Structure 4-2
Output 4-2
JSON Structure 4-3
Table Impact 4-3
Clearance Induction Service 4-3
Service Type 4-3
REST URL 4-3
Input Parameters 4-4
JSON Structure 4-4
Output 4-4
JSON Structure 4-5
Table Impact 4-5

5 Backend System Administration and Configuration

Supported Environments 5-1
Exception Handling 5-1
Logging Configuration 5-1
ADF Logging 5-2
Batch Client Logging 5-2
Batch_logging Properties 5-2
Configurable GTTCapture 5-3

Send Us Your Comments

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

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

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

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

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

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

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

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

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

Preface

This *Oracle Retail Pricing Cloud Service Operations Guide* provides critical information about the processing and operating details of Product, including the following:

Audience

This guide is for:

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

Documentation Accessibility

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Retail Price Management Release 16.0.030 documentation set:

- *Oracle Retail Price Management Release Notes*
- *Oracle Retail Price Management Installation Guide*
- *Oracle Retail Price Management User Guide*
- *Oracle Retail Merchandising Operations Management Implementation Guide*
- *Oracle Retail Merchandising Operations Management Batch Schedule*

Customer Support

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

https://support.oracle.com

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 16.0) or a later patch release (for example, 16.0.1). If you are installing the base release and additional patch releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch releases can contain critical information related to the base release, as well as information about code changes since the base release.

Improved Process for Oracle Retail Documentation Corrections

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

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

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

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

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

Oracle Retail Documentation on the Oracle Technology Network

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

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

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

Conventions

The following text conventions are used in this document:

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

1
Introduction

Oracle Retail Pricing Cloud Service (RPCS) provides the ability to define, maintain, and review price changes, clearances, and promotions as well as provides the ability to pass approved price events onto downstream selling systems.

RPCS functionality provides support for initial pricing, regular price changes, clearance markdowns, and promotions. It also offers the ability to upload price changes and clearance events in multiple ways including spreadsheet induction, bulk upload, or web service. The user interface supports price change or clearance wizards for entering multiple price events at once as well as a quick entry panel for quickly adding these types of price events.

RPCS also supports an offer wizard for entering and maintaining offers within a promotion. Offers can be at the item or transaction level and RPCS supports multiple templates for different offer types.

RPCS functionality provides support for initial pricing, regular price changes, clearance markdowns, and promotions. It also provides for the execution of these price events to update the selling systems, such as Xstore POS, and to update the item/location price and stock ledger when the price changes go into effect.

Note: Users should not access the Oracle Retail Pricing Cloud Service during the Retail Merchandising System (RMS) batch window as it may cause some unpredictable results.

2

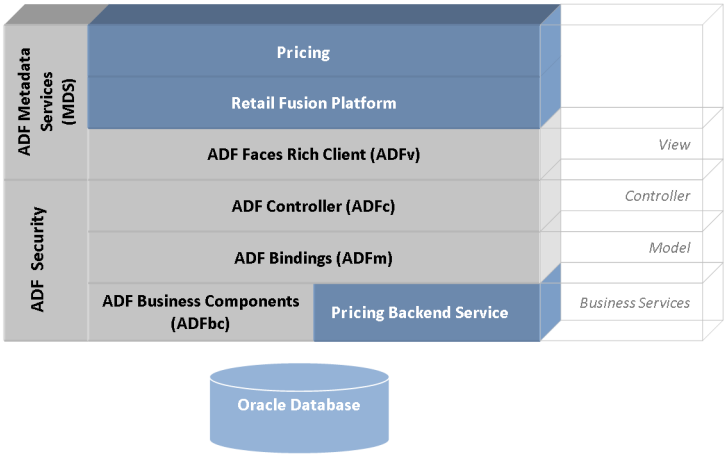
Technical Architecture

This chapter describes the overall software architecture for Oracle Retail Pricing Cloud Service. The chapter provides a high-level discussion of the general structure of the system, including the various layers of Java code.

Overview

Retail Applications are based on the Oracle Application Development Framework (ADF). The following diagram shows the key components that make up the architecture of Retail Applications.

Figure 2–1 Oracle Retail Pricing Cloud Service n-tier Architecture



Oracle Application Development Framework (ADF)

Oracle Application Development Framework (ADF) supports organizations in building cutting-edge rich enterprise business applications that can be customized and personalized in all dimensions. Customizations are global changes, visible to all users that are performed by an administrator. Personalization are user-made changes that are only visible to the person making the change.

ADF is based on the Java Enterprise Edition platform.

Model-View-Controller (MVC) Architectural Pattern

Applications built using ADF follow a Model-View-Controller (MVC) architectural pattern. The goal of the MVC pattern is to clearly separate the application's functionality into a set of cooperating components.

ADF provides a set of components that realize the goals of each part of MVC pattern.

- Model is realized by the ADF Bindings Layer.
- Controller is realized by the ADF Controller Layer.
- View is realized by the ADF Faces Layer.
- ADF Business components and other backend components that sit below the Model layer are called Business Services.

ADF Security

The ADF security layer provides the following:

- Standards based (Oracle Platform Security Services (OPSS)) security framework with default roles and permissions.
- Tools to generate file-based identity store (for both Oracle Internet Directory and AD) based on the framework.
- Tools to migrate file-based security store in to database for QA and production environments.
- Reference implementation for clients to manage the security based on their business needs.
- OPSS-based batch security framework (Retail Fusion Platform).
- Tools/documentation to implement centralized logout in Single Sign-On (SSO) (Oracle Access Management (OAM)) environments.

ADF View (ADFv)

The View layer provides the user interface to the application. The view layer uses HTML, rich Java components or XML and its variations to render the user interface. JSF based tag libraries are used for displaying the UI.

ADF Controller (ADFc)

The ADF Controller layer controls the application's flow. Web based applications are composed of multiple web pages with dynamic content. The controller layer manages the flow between these pages. Different models can be used when building this later. The most prominent architecture for Java-based web applications relies on a servlet that acts as the controller. The Apache Jakarta Struts controller, an open source framework controller, is the de facto standard for Java-based web systems. Oracle ADF uses the Struts controller to manage the flow of web applications.

ADF Business Components (ADFbc)

The business service layer manages the interaction with a data persistence layer. It provides services as data persistence, object/relational mapping, transaction management and business logic execution.

Business Components easily map the database object and extend it with business logic, validation and so on.

The idea behind Business Components is to abstract the data layer from the view layer. This is a key concept in the MVC pattern. Business Components will expose the interface to the view layer by using an application module that contains View Object. Those view objects contain a specific usage of the data layer.

ADF Business Components implements the business service through the following set of cooperating components:

- Entity object – An entity object represents a row in a database table and simplifies modifying its data by handling all data manipulation language (DML) operations for you. It can encapsulate business logic for the row to ensure that your business rules are consistently enforced. You associate an entity object with others to reflect relationships in the underlying database schema to create a layer of business domain objects to reuse in multiple applications.
- View object – A view object represents a SQL query. You use the full power of the familiar SQL language to join, filter, sort, and aggregate data into exactly the shape required by the end-user task. This includes the ability to link a view object with others to create master-detail hierarchies of any complexity. When end users modify data in the user interface, view objects collaborate with entity objects to consistently validate and save the changes.
- Application module – An application module is the transactional component that UI clients use to work with application data. It defines an updatable data model and top-level procedures and functions (called service methods) related to a logical unit of work related to an end-user task.

ADF Model (ADFm)

This component acts as the connector between the view and business logic layers.

The Model layer connects the Business Services to the objects that use them in the other layers. Oracle ADF provides a Model layer implementation that sits on top of Business Services, providing a single interface that can be used to access any type of Business Services.

Developers get the same development experience when binding any type of Business Service layer implementation to the view and Controller layers. The Model layer in Oracle ADF served as the basis for JSR 227, A Standard Data binding & Data Access Facility for J2EE.

Oracle Metadata Services (MDS)

The ability of an application to adapt to changes is a necessity that needs to be considered in the application design and that should drive the selection of the development platform and architecture. Flexible business applications must be able to adapt to organizational changes, different end user preferences and changes in the supported business are required.

MDS is the customization and personalization framework integral to Oracle Fusion Middleware and a key differentiator of the Oracle development platform. MDS provides a repository for storing metadata for applications, such as customizations and persisted personalization files and configurations.

Retail Applications allow the following through MDS:

- Personalization of saved searches through MDS.
- Implicit personalization of few ADF UI attributes.

Retail Fusion Platform

The Retail Fusion Platform (commonly referred to as Platform) is a collection of common, reusable software components that serve as foundation for building Oracle Retail's next generation ADF-based applications. The Platform imposes standards and patterns along with a consistent look and feel for Oracle Retail's ADF applications.

RPCS Backend Service

RPCS Backend Services are collective Business Legacy components that are reused in this ADF version of RPCS.

Data Access Patterns

Database interaction between the middle tier and the database is done using the industry standard Java Database Connectivity Protocol (JDBC). JDBC facilitates the communication between a Java application and a relational database.

Database Access Using ADFbc

JDBC is engrained within Oracle ADF Business Components as the primary mechanism for its interaction between the middle tier and the database. SQL is realized within ADF business components to facilitate create, read, update and delete (CRUD) actions.

Connection Pooling

When the application 'disconnects' a connection, the connection is saved into a pool instead of being actually disconnected. A standard connection pooling technique, this saved connection enables Retail Applications to reuse the existing connection from a pool. In other words, the application does not have to complete the connection process for each subsequent connection.

Data Storage

The Oracle Database realizes the database tier in a Retail Application's architecture. It is the application's storage platform, containing the physical data (user and system) used throughout the application. The database tier is only intended to handle the storage and retrieval of information and is not involved in the manipulation or in the delivery of the data. This tier responds to queries; it does not initiate them.

Accessing Merchandising System Data in Real Time

The data that Retail Application utilizes is located in both application-specific tables and merchandising system (RMS, for example) tables. Because Retail Applications share the same schema as the merchandising system (RMS, for example), the application is able to interact with the merchandising system's data directly, in real time.

3

Pricing Batch Processes

This chapter discusses Java-based batch processing within RPM.

Table 3–1 Functional Descriptions and Dependencies

Batch processes	Details
BDI Clearance Publishing	This batch process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to stage clearance data. The batch job BDI_ PRICING_CLR_TX _JOB is defined in the Merchandising JOS batch job admin to stage clearance data.
BDI Price Change Publishing	This batch process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to stage clearance data.The batch job BDI_ PRICING_PC_TX _JOB is defined in the Merchandising JOS batch job admin to stage clearance data.
Promotion Publishing (BDI_ PRICING_PROMO_OFFER_TX_ CYCLE_JOB)	This batch process utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to target applications. The batch job BDI_PRICING_ PROMO_OFFER_TX_CYCLE_JOB is defined in the Merchandising JOS batch job admin to stage promotions data.
ClearanceInductionBatch	This batch program allows the user to upload clearance events in bulk.
ClearancePriceChangePublishBatch	This batch process formats and stages output of clearance price change price events to be published via a flat file format.
futureRetailPurgeBatch	This timed multi-threaded batch deletes records from future retail tables that are past the retention period of the associated price events.
FutureRetailRollUpBatch.sh	This batch attempts to roll up timelines at a lower level by comparing lower level timelines to higher levels and removing any lower level timelines that match higher level timelines exactly.
itemReclassBatch	When items are moved from one department/class/subclass to another in the merchandising system, this batch process accordingly sets the correct department/class/subclass for these items in the RPM_FUTURE_ RETAIL table.
NewItemLocationBatch	This batch ranges item locations by putting them into the future retail table and RPM_ITEM_LOC. Item and locations are fed to this program via the RPM_ITEM_LOC_WS table, which is populated by an RMS process.
NightlyBatchCleanup	This batch performs "clean up" logic against Pricing database objects.
PriceChangeInductionBatch	This batch program allows the user to upload regular price changes in bulk.
PriceEventExecutionBatch	This batch process performs the necessary work to start (regular price change, clearance price change, promotions) and end (price clearances, promotions) pricing events.

Table 3–1 (Cont.) Functional Descriptions and Dependencies

Batch processes	Details
PurgeBatch	This generic purge batch calls most of the purge batches into one purge process.
PurgeGttCaptureBatch	This batch process deletes records from gtt data capture tables.
RegularPriceChangePublishBatch	This batch process formats and stages output of regular price change price events.
RefreshPosDataBatch	The RefreshPosDataBatch program deletes the contents of the payload tables.

BDI Clearance Publishing (BDI_PRICING_CLR_TX _JOB)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to target applications. The batch job BDI_PRICING_CLR_TX _JOB is defined in the Merchandising JOS batch job admin to stage clearance data.

The program can be scheduled ad hoc, recurring, and nightly depending on the needs of each retailer. Each run of the program will include changes for approved clearances since the previous run.

Scheduling Constraints

Table 3–2 BDI_PRICING_CLR_TX_JOB Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring, Nightly
Scheduling Considerations	not applicable
Pre-Processing	not applicable
Post-Processing	not applicable
Threading Scheme	not applicable

Restart/Recovery

N/A

Key Tables Affected

Table 3–3 Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_CLR_BDI_HELPER_WS	Yes	No	No	No
RPM_CLEARANCE	Yes	No	No	No
RPM_PRICE_EVENT_PAYLOAD	Yes	No	No	No

Design Assumptions

N/A

Output

BDI extractor jobs call respective BDI functions to extract data from Pricing tables to BDI outbound staging table CLEARANCE_OUT.

Table 3–4 BDI Outbound Staging Table CLEARANCE_OUT

Name	Null	Type	Description
BDI_SEQ_ID	No	NUMBER	BDI Internal Column
BDI_APP_NAME	No	VARCHAR2(50)	BDI Internal Column
BDI_DATASET_TYPE	Yes	VARCHAR2(20)	BDI Internal Column
BDI_DATASET_ACTION	Yes	VARCHAR2(20)	BDI Internal Column
REC_ID	No	NUMBER(10,0)	The ID of the record
RECORD_TYPE	No	VARCHAR2(50)	The record type. Valid values (Create/Update/Delete)
CLEARANCE_ID	No	NUMBER(15,0)	The clearance ID
ITEM	Yes	VARCHAR2(25)	The item ID
LOCATION	Yes	NUMBER(10,0)	The location ID
LOCATION_TYPE	Yes	VARCHAR2(30)	The location Type. Valid values (S 'Store' or W 'Warehouse).
EFFECTIVE_DATE	Yes	TIMESTAMP(2)	Effective date of the clearance
RETAIL	Yes	NUMBER(20,4)	The clearance retail for the item location
UOM	Yes	VARCHAR2(25)	The retail unit of measure
CURRENCY	Yes	VARCHAR2(25)	The currency for the location
RESET_INDICATOR	No	NUMBER(1,0)	Indicates if the clearance event is a reset. Valid values: 0- The record is not a reset; 1 – The record is a reset.

BDI Price Change Publishing (BDI_PRICING_PC_TX_CYCLE_JOB)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to target applications. The batch job BDI_PRICING_PC_TX _JOB is defined in the Merchandising JOS batch job admin to stage clearance data.

The program can be scheduled ad hoc, recurring, and nightly depending on the needs of each retailer. Each run of the program will include changes for approved price changes since the previous run.

Scheduling Constraints

Table 3–5 BDI_PRICING_PC_TX_CYCLE_JOB Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring, Nightly
Scheduling Considerations	not applicable
Pre-Processing	not applicable
Post-Processing	not applicable
Threading Scheme	not applicable

Restart/Recovery

N/A

Key Tables Affected

Table 3–6 Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_PC_BDI_HELPER_WS	Yes	No	No	No
RPM_PRICE_EVENT_PAYLOAD	Yes	No	No	No
RPM_PRICE_CHANGE	Yes	No	No	No

Design Assumptions

N/A

Output

BDI extractor jobs call respective BDI functions to extract data from Pricing tables to BDI outbound staging table PRICE_CHANGE_OUT.

Table 3–7 BDI Outbound Staging Table PRICE_CHANGE_OUT

Name	Null	Type	Description
BDI_SEQ_ID	No	NUMBER	BDI Internal Column
BDI_APP_NAME	No	VARCHAR2(50)	BDI Internal Column
BDI_DATASET_TYPE	Yes	VARCHAR2(20)	BDI Internal Column
BDI_DATASET_ACTION	Yes	VARCHAR2(20)	BDI Internal Column
REC_ID	No	NUMBER(10,0)	The ID of the record
RECORD_TYPE	No	VARCHAR2(50)	The record type. Valid values (Create/Update/Delete)
PRICE_CHANGE_ID	No	NUMBER(15,0)	The price change ID
ITEM	Yes	VARCHAR2(25)	The item ID
LOCATION	Yes	NUMBER(10,0)	The location ID
LOCATION_TYPE	Yes	VARCHAR2(30)	The location Type. Valid values (S ‘Store’ or W ‘Warehouse)
EFFECTIVE_DATE	Yes	TIMESTAMP(2)	Effective date of the price change
RETAIL	Yes	NUMBER(20,4)	The new regular retail for the item location
UOM	Yes	VARCHAR2(25)	The retail unit of measure
CURRENCY	Yes	VARCHAR2(25)	The currency for the location
RETAIL_CHANGE_IND	No	NUMBER(6,0)	Indicates whether the retail changed with this price change

Promotion Publishing (BDI_PRICING_PROMO_OFFER_TX_CYCLE_JOB)

This program utilizes BDI (Bulk Data Integration) to facilitate the bulk data movement to target applications. The batch job BDI_PRICING_PROMO_OFFER_TX_CYCLE_JOB is defined in the Merchandising JOS batch job admin to stage promotions data.

The program can be scheduled ad hoc, recurring, and nightly depending on the needs of each retailer. Each run of the program will include changes for approved price changes since the previous run.

Scheduling Constraints

Table 3–8 BDI_PRICING_CLR_TX_JOB Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring, Nightly
Scheduling Considerations	not applicable
Pre-Processing	not applicable
Post-Processing	not applicable
Threading Scheme	not applicable

Restart/Recovery

N/A

Promotions Integration

The Pricing Cloud Service will communicate promotional offers at a rule criteria level instead of the item/location level, where applicable. Offers in an approved or active state and candidates for integration when edited.

Table 3–9 has some examples of message types that are created when certain activities are being triggered by user:

Table 3–9 Promotions Integration Examples

Message Type	Activities in Promotion UI
OfferCreate	This message type is generated when the user changes the state of an Offer to Approved state.
OfferDelete	This message type is generated when the following action happen: ■ The user changes the state of an Offer to Delete state ■ The user deletes an Offer
OfferHeaderUpdate	This message type is generated when the user updates the Promotion Offer Header information that is already in Approved state, for example, changing the start date, end date, comments, or coupon code of the promotion.
OfferCondUpdate	This message type is generated when the user updates the Condition information of an Offer, for example, changing the spending type or the spending value.
OfferRwdUpdate	This message type is generated when the user updates the Reward information of an Offer, for example, the reward value (change_type, change_amount, change_percent, etc.) of a promotion.
OfferRwdMerchAdd	This message type is generated when the user adds merchandise hierarchy to a promotion Reward list.

Table 3–9 (Cont.) Promotions Integration Examples

Message Type	Activities in Promotion UI
OfferCondMerchAdd	This message type is generated when the user adds merchandise hierarchy to a promotion Condition list.
OfferCondMerchDel	This message type is generated when the user deletes merchandise hierarchy from Promotion Condition list.
OfferRwdMerchDel	This message type is generated when the user deletes merchandise hierarchy from Promotion Reward list.
OfferLocAdd	This message type is generated when the user adds a location to a promotion.
OfferLocDel	This message type is generated when the user deletes a location to a promotion.
OfferCancel	This message type is generated when the user cancel a Promotion Offer.
OfferCondMerchCancel	This message type is generated when the user cancel a merchandise hierarchy from the Condition List of Promotion Offer.
OfferRwdMerchCancel	This message type is generated when the user cancel a merchandise hierarchy from the Reward List of Promotion Offer.
OfferLocCancel	This message type is generated when the user cancel a location of Promotion Offer.

Payload Tables

Table 3–10 lists Payload tables are used to hold staged data for BDI population:

Table 3–10 Payload Tables that Hold Staged Data for BDI population

Table	Description
RPM_PRICE_EVENT_PAYLOAD	oMessage header level data - shared with PC and CLR Payload data. For promotion offers, the RIB_TYPE field will hold the same values as what is staged in the RPM_PROMO_OFFER_PUB_WS.PUB_TYPE_CODE table.
RPM_PROMO_OFFER_PUB_WS.PUB_TYPE_CODE	This table is populated for all promotion offer messages.
RPM_PROMO_OFFER_PAYLOAD	Offer header level data. This table is populated for all promotion offer messages.
RPM_PROMO_OFR_CND_MRCH_PAYLOAD	This table holds the payload information of the merchandise nodes associated with a promotion offer condition. This table is only populated when the condition for an approved offer has new merchandise data added to it or deleted from it.
RPM_PROMO_OFR_RWD_MRCH_PAYLOAD	This table holds the payload information of the merchandise nodes associated with a promotion offer reward. This table is only populated when the reward for an approved offer has new merchandise data added to it or deleted from it.
RPM_PROMO_OFFER_LOC_PAYLOAD	This table holds the payload information of the location nodes associated with a promotion offer. Only store data will be on this table – any zones associated with an offer will be exploded out to store level. This table is only populated when an approved offer has new zone/loc data added to it or deleted from it.

Table 3–10 (Cont.) Payload Tables that Hold Staged Data for BDI population

Table	Description
RPM_PROM_OFR_ CNCL_ITEM_PAYLOAD	This table holds the payload information for items cancelled from active promotion offers. This table is only populated for active offers when merchandise is cancelled from a reward or condition.
RPM_PROM_OFR_ CNCL_LOC_PAYLOAD	This table holds the payload information for locations cancelled from active promotion offers.
RPM_PROMO_OFFER_ CANCEL_PAYLOAD	This table holds the payload information for when active promotion offers are cancelled as a whole.

Payload Population Logic

In many situations, the payload population logic will only populate a small piece of data about an offer (the action along with the promotion and offer ids). In these situations, the BDI extraction logic will use the promotion offer operational tables as the main source of data to stage in the BDI tables rather than the payload tables. In all scenarios, the RPM_PRICE_EVENT_PAYLOAD and RPM_PROMO_OFFER_ PAYLOAD tables will be utilized.

BDI data population retrieves the data from both the Payload tables and also Promotion Operational tables depending on the message type.

In order to populate the data into BDI tables more efficiently, a workspace tables called RPM_PROMO_BDI_HELPER_WS is used. This workspace table will be inserted with the data from the two main payload tables that drives the whole processes, rpm_price_ event_payload and rpm_promo_offer_payload tables.

Once the workspace table is populated, the system will perform some cleanup activities so that if the same promotions are being updated more than once, it will only write one Update message with the latest information into BDI tables. If the Promotion is being created and then Deleted, it will not get written into BDI tables at all. In order to avoid multiple update messages under the same promotion as mentioned previously, the system will retrieve the data from the Promotion Operational tables itself in order to get the latest data set of a promotion. The only time that payload tables records any activities is when a Promotion is deleted from the system, any merchandise hierarchy is deleted from a Promotion Condition or Reward, location is deleted from promotion and Canceling Promotion. Once the cleanup processed is done, the data will be inserted into BDI tables.

Table 3–11 maps what message structures use which payload or operational tables to build the BDI data:.

Table 3–11 Message Structures to Build the BDI data

Message Type	Source Tables	Target Tables
OfferCreate	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER, RPM_PROMO_OFFER_COND, RPM_PROMO_OFFER_COND_MERCH, RPM_PROMO_OFFER_REWARD, RPM_PROMO_OFFER_REWARD_MERCH, RPM_PROMO_OFFER_ZONE_LOC	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_OUT, PROMO_OFFER_COND_OUT, PROM_OFR_CND_MRCH_OUT, PROMO_OFR_REWARD_OUT, PROM_OFR_RWD_MRCH_OUT, PROMO_OFFER_LOC_OUT
OfferDelete	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_OUT
OfferUpdate	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_OUT
OfferCondUpdate	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_COND	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_COND_OUT
OfferCondMerchAdd	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_COND_MERCH	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_CND_MRCH_OUT
OfferRwdUpdate	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_REWARD	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFR_REWARD_OUT
OfferRwdMerchAdd	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_REWARD_MERCH	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_RWD_MRCH_OUT
OfferCondMerchDel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFR_CND_MRCH_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_CND_MRCH_OUT
OfferRwdMerchDel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFR_CND_MRCH_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_RWD_MRCH_OUT
OfferLocAdd	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_LOC_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_LOC_OUT
OfferLocDel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_LOC_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFFER_LOC_OUT
OfferCancel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROMO_OFFER_CANCEL_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROMO_OFR_CANCEL_OUT
OfferCondMerchCancel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROM_OFR_CNCL_ITEM_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_CNCL_ITM_OUT
OfferRwdMerchCancel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROM_OFR_CNCL_ITEM_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_CNCL_ITM_OUT
OfferLocCancel	RPM_PRICE_EVENT_PAYLOAD, RPM_PROMO_OFFER_PAYLOAD, RPM_PROM_OFR_CNCL_LOC_PAYLOAD	PRC_PAYLD_MSG_HDR_OUT, PROM_OFR_CNCL_LOC_OUT

BDI Tables

The message structure in the BDI tables will that even though there is a hierarchical relationship between tables, each table will not reference an ID for the entity above it. Rather, each full message will be identified by a unique ID that is stored on all tables – the PAYLOAD_ID column on each BDI table. Thus, for an OfferCreate message, each of the seven BDI tables populated will have data with the same PAYLOAD_ID value and the consuming system will need to understand the structure of the tables in relation to each other for that specific message type.

Table 3–12 lists the BDI tables and its parent table from a high level perspective

Table 3–12 BDI Tables

BDI Table	Parent Table
PRC_PAYLD_MSG_HDR_OUT	none
PROMO_OFFER_OUT	PRC_PAYLD_MSG_HDR_OUT
PROMO_OFFER_COND_OUT	PROMO_OFFER_OUT
PROMO_OFR_REWARD_OUT	PROMO_OFFER_OUT
PROM_OFR_CND_MRCH_OUT	PROMO_OFFER_COND_OUT
PROM_OFR_RWD_MRCH_OUT	PROMO_OFR_REWARD_OUT
PROMO_OFFER_LOC_OUT	PROMO_OFFER_OUT
PROMO_OFR_CANCEL_OUT	PRC_PAYLD_MSG_HDR_OUT
PROM_OFR_CNCL_ITM_OUT	PRC_PAYLD_MSG_HDR_OUT
PROM_OFR_CNCL_LOC_OUT	PRC_PAYLD_MSG_HDR_OUT

Key Tables Affected

Table 3–13 BDI Key Tables Affected

Table	Select	Insert	Update	Delete
PRC_PAYLD_MSG_HDR_OUT	Yes	No	No	No
PROMO_OFFER_OUT	Yes	No	No	No
PROMO_OFFER_OUT	Yes	No	No	No
PROMO_OFFER_COND_OUT	Yes	No	No	No
PROMO_OFR_REWARD_OUT	Yes	No	No	No
PROMO_OFFER_OUT	Yes	No	No	No
PRC_PAYLD_MSG_HDR_OUT	Yes	No	No	No
PROMO_OFFER_OUT	No	Yes	No	No
PROMO_OFFER_COND_OUT	No	Yes	No	No
PROMO_OFR_REWARD_OUT	No	Yes	No	No
PROM_OFR_CND_MRCH_OUT	No	Yes	No	No
PROM_OFR_RWD_MRCH_OUT	No	Yes	No	No
PROMO_OFFER_LOC_OUT	No	Yes	No	No

Table 3–13 (Cont.) BDI Key Tables Affected

Table	Select	Insert	Update	Delete
PROMO_OFR_CANCEL_OUT	No	Yes	No	No
PROM_OFR_CNCL_ITM_OUT	No	Yes	No	No
PROM_OFR_CNCL_LOC_OUT	No	Yes	No	No

Output

BDI extractor jobs call respective BDI functions to extract data from Pricing tables to BDI outbound staging tables which mirror the structure of the Pricing promotions tables.

PRC_PAYLD_MSG_HDR_OUT

Table 3–14 PRC_PAYLD_MSG_HDR_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PRC_PAYLD_MSG_HDR_ID	NUMBER(10)	No	The unique payload ID for the message.
MESSAGE_TYPE	VARCHAR2(50)	No	The type of message associated to the payload_id.

PROMO_OFFER_OUT

Table 3–15 PROMO_OFFER_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROMO_OFFER_ID	NUMBER(10)	No	The payload ID of the promotion offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
OFFER_DESC	VARCHAR2(1000)	Yes	Offer description.
OFFER_CUST_DESC	VARCHAR2(1000)	Yes	The customer description of the offer.
LEVEL_CODE	NUMBER(2)	Yes	The level of the offer. Valid values are: 0 - Item, 1 - Transaction.
TYPE_CODE	NUMBER(2)	Yes	The type of the offer. Valid values are: 0 Item Simple, 1 Transaction Simple, 2 - Transaction Buy Get

Table 3–15 (Cont.) PROMO_OFFER_OUT

COLUMN	TYPE	NULLABLE	COMMENT
TEMPLATE_ID	NUMBER(2)	Yes	The template of the offer. Valid values are: 0 - Get Discount, 1 - Buy X Get Discount, 2 - Spend X Get Discount, 4 - Get Y For Discount
START_DATE	TIMESTAMP(6)	Yes	The start date and time of the offer.
END_DATE	TIMESTAMP(6)	Yes	The end date and time of the offer.
COMMENTS	VARCHAR2(4000)	Yes	The comments for the offer.
COUPON_CODE	VARCHAR2(160)	Yes	The coupon code for the offer.

PROMO_OFFER_COND_OUT

Table 3–16 PROMO_OFFER_COND_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROMO_OFFER_COND_ID	NUMBER(10)	No	The payload ID of the condition of a promotion offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
COND_ID	NUMBER(15)	No	Condition ID.
BUY_SPEND_TYPE	NUMBER(1)		The buy spend type of the condition. Valid values are: 0 - Quantity, 1 - Amount
BUY_SPEND_VALUE	NUMBER(20, 4)	No	The buy spend value of the condition.
BUY_UOM	VARCHAR2(4)	Yes	The buy UOM of the condition.

PROMO_OFR_REWARD_OUT

Table 3–17 PROMO_OFR_REWARD_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROMO_OFR_REWARD_ID	NUMBER(10)	No	The payload ID of the reward of a promotion offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.

Table 3–17 (Cont.) PROMO_OFR_REWARD_OUT

COLUMN	TYPE	NULLABLE	COMMENT
OFFER_ID	NUMBER(10)	No	The offer ID.
REWARD_ID	NUMBER(15)	No	Reward ID.
CHANGE_TYPE	NUMBER(1)	No	Type of change for the reward. Valid values: change by amount (1), change by percent (0), fixed price (2)
CHANGE_AMOUNT	NUMBER(20,4)	Yes	The change by amount or fixed price amount.
CHANGE_PERCENT	NUMBER(7,4)	Yes	Percentage value when change type is change by percent.
QTY_TO_DISC	NUMBER(7,4)	Yes	The quantity to discount.
QTY_TO_DISC_UOM	VARCHAR2(4)	Yes	UOM of the discount quantity.
APPLY_TO_IND	NUMBER(1)	No	The apply to indicator of the reward. Valid values: Regular only - 0; Clearance only - 1; Regular and Clearance - 2

PROM_OFR_CND_MRCH_OUT

Table 3–18 PROM_OFR_CND_MRCH_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROM_OFR_CND_MRCH_ID	NUMBER(10)	No	The payload ID of the condition of a promotion offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
COND_ID	NUMBER(15)	No	Condition ID.
MERCH_LVL	NUMBER(2)	No	The merchandise level. Valid values are: 1 - Department; 2 - Class; 3 - Subclass; 4 - Parent Item; 5 - Parent/Diff Item; 6 - Transaction Item; 8 - All Departments
DEPT	NUMBER(4)	Yes	Department ID.
CLASS	NUMBER(4)	Yes	Class ID.
SUBCLASS	NUMBER(4)	Yes	Subclass ID.
ITEM	VARCHAR2(25)	Yes	Item.
DIFF_ID	VARCHAR2(10)	Yes	Differentiator ID.
EXCLUDE_IND	NUMBER(1)	No	The exclude indicator.

PROMO_OFFER_LOC_OUT

Table 3–19 PROMO_OFFER_LOC_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROMO_OFFER_LOC_ID	NUMBER(10)	No	The payload ID of the location node associated with the promotion offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
LOCATION	NUMBER(10)	Yes	Location for the offer.
EXCLUDE_IND	NUMBER(1)	No	The exclude indicator.

PROMO_OFR_CANCEL_OUT

Table 3–20 PROMO_OFR_CANCEL_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROMO_OFR_CANCEL_ID	NUMBER(10)	No	The payload ID of the offer cancellation.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
CANCEL_DATETIME	TIMESTAMP(6)	No	The date and time that the offer cancellation takes effect.

PROM_OFR_CNCL_ITM_OUT

Table 3–21 PROM_OFR_CNCL_ITM_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROM_OFR_CNCL_ITM_ID	NUMBER(10)	No	The payload ID of the item cancellation from the offer.

Table 3–21 (Cont.) PROM_OFR_CNCL_ITM_OUT

COLUMN	TYPE	NULLABLE	COMMENT
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
REWARD_COND_IND	VARCHAR2(1)	No	The date and time that the offer cancellation takes effect.
COND_ID	NUMBER(15)	Yes	Condition ID.
REWARD_ID	NUMBER(15)	Yes	Reward ID.
MERCH_LVL	NUMBER(2)	No	The merchandise level. Valid values are: 1 - Department; 2 - Class; 3 - Subclass; 4 - Parent Item; 5 - Parent/Diff Item; 6 - Transaction Item; 8 - All Departments
DEPT	NUMBER(4)	Yes	Department ID.
CLASS	NUMBER(4)	Yes	Class ID.
SUBCLASS	NUMBER(4)	Yes	Subclass ID.
ITEM	VARCHAR2(25)	Yes	Item.
DIFF_ID	VARCHAR2(10)	Yes	Differentiator ID.
CANCEL_DATETIME	TIMESTAMP(6)	No	The date and time that the offer cancellation takes effect.

PROM_OFR_CNCL_LOC_OUT

Table 3–22 PROM_OFR_CNCL_LOC_OUT

COLUMN	TYPE	NULLABLE	COMMENT
BDI_SEQ_ID	NUMBER	No	bdi internal column
BDI_APP_NAME	VARCHAR2(50)	No	bdi internal column
BDI_DATASET_TYPE	VARCHAR2(20)	Yes	bdi internal column
BDI_DATASET_ACTION	VARCHAR2(20)	Yes	bdi internal column
PROM_OFR_CNCL_LOC_ID	NUMBER(10)	No	The payload ID of the location cancellation from the offer.
PAYLOAD_ID	NUMBER(10)	No	The message payload ID.
PROMO_ID	NUMBER(10)	No	The promo ID.
OFFER_ID	NUMBER(10)	No	The offer ID.
LOCATION	NUMBER(10)	Yes	Location cancelled from the offer.
CANCEL_DATETIME	TIMESTAMP(6)	No	The date and time that the offer cancellation takes effect.

ClearanceInductionBatch (Clearance Induction Batch)

Table 3–23 ClearanceInductionBatch Details

Module Name	ClearanceInductionBatch.sh
Description	Clearance bulk upload process
Functional Area	Clearance
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	ClearanceInductionBatch.sh <user alias> <incoming-dir-path> <Template_Key> [filter_Str] 1. The first argument is the user alias name which mandatory and is mapped to an authorized user in the system. 2. The second argument is the path where the induction input files are placed. 3. Third argument is the name of the standard clearance template key. This is a mandatory argument. 4. The fourth argument is an optional and when provided accepts the following values: a. XML - indicates that the batch has to look for xml files in the given incoming-dir-path and process them. b. ZIP - This is the default value when nothing is provided. The process will look for zip files containing xml files and process them sequentially.

Note: File naming standards

XML file:

The file should have a prefix of CLIND. Ex: CLIND_ABC-10.10.18.xml

The file should contain the data in the format suggested by standard clearance upload template.

ZIP file:

The file should have a prefix of CLIND. Ex: CLIND_ABC.ZIP

The xml files with in the zip file should also have the prefix of CLIND.

Design Overview

The clearance induction batch process perform the necessary work to upload clearances in bulk. For the bulk upload, clearance data will be present in XML format with the data formatted in the standard clearance upload template. This batch accepts the clearance data present in XML format and also as zip files of xml files formatted in the standard template.

Scheduling Constraints

Table 3–24 ClearanceInductionBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 3–25 ClearanceInductionBatch Key Tables Affected

Table	Select	Insert	Update	Delete
S9T_TEMPLATE	Yes	No	No	No
SVC_PROCESS_TRACKER	Yes	No	No	No
S9T_ERRORS	Yes	Yes	Yes	No
RPM_CORESVC_CLEARANCE_ERR	Yes	No	No	No
RPM_SVC_CLEARANCE	Yes	No	Yes	No
RPM_CLEARANCE	Yes	No	No	No
RPM_CLEARANCE_GROUP	No	Yes	No	No

Design Assumptions

N/A

ClearancePriceChangePublishBatch (Clearance Price Change Publish Batch)

Table 3–26 ClearancePriceChangePublishBatch Details

Module Name	ClearancePriceChangePublishBatch.sh
Description	Clearance events are exported
Functional Area	Clearance
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	ClearancePriceChangePublishBatch.sh <user_alias> <outgoing-dir-path>

Design Overview

The ClearancePriceChangePublishBatch program formats and stages output of clearance price change price events.

The corresponding clearancePriceChangePublishExport shell script produces a pipe ("|") delimited flat-file export based on the output of the ClearancePriceChangePublishBatch.

The batch looks for price events in the RPM_PRICE_EVENT_PAYLOAD table with a RIB_FAMILY of 'ClrPrcChg' and distributes those events to multiple threads based on the settings in the RPM_BATCH_CONTROL table. Each thread reads in its set of clearance price change events from tables RPM_PRICE_EVENT_PAYLOAD and RPM_CLEARANCE_PAYLOAD and generates output in RPM_PRICE_PUBLISH_DATA. After the flat file is successfully generated by the Export script (see the following format), the associated records in the payload tables are deleted.

Then the flat-files per location based on the data from payload table that need to be published/processed will be created and zipped and copied to the outgoing-dir-path provided as a batch parameter.

Scheduling Constraints

Table 3–27 ClearancePriceChangePublishBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	The ClearancePriceChangePublishBatch program is threaded, using RPM_BATCH_CONTROL. The LUW is a single clearance price change event.

Restart/Recovery

N/A

Key Tables Affected

Table 3–28 ClearancePriceChangePublishBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_PRICE_EVENT_PAYLOAD	Yes	No	No	No
RPM_CLEARANCE_PAYLOAD	Yes	No	No	No

Output File

FHEAD - REQUIRED: File identification, one line per file.

FDETL - OPTIONAL: Price Change Event (Create or Modify).

FDELE - OPTIONAL: Price Change Event (Delete).

FTAIL - REQUIRED: End of file marker, one line per file.

Note: File naming standards

The naming convention for the flat file will be (CLRPC_<timestamp>_<location>_<loc_type>.dat), where <timestamp> is the current system time stamp, <location> is the location ID and <loc_type> is the type of the location where 'S' is for Store and 'W' is for Warehouse. The zip file naming convention will be (CLRPC_<timestamp>.zip).

Output File Layout

Table 3–29 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Line ID	Number(10)	1	Unique line identification
	File Type	Char(5)	CLRPC	Clearance Price Changes
	Export timestamp	Timestamp		System clock timestamp (YYYYMMDDHHMISS)
	Location	Number(10)		Location identifier
	Location Type	Char(1)		S = Store, W = Warehouse
FDETL	Record Descriptor	Char(5)	FDETL	File Detail Marker (1 per clearance create or modify)
	Line ID	Number(10)		Unique line identification
	Event Type	Char(3)		CRE = Create, MOD = Modify
	Id	Number(15)		Clearance identifier
	Item	Char(25)		Item identifier
	Effective Date	Date		Clearance Effective Date (YYYYMMDDHH24MISS)
	Selling Retail	Number(20,4)		Selling retail with price change applied
	Selling Retail UOM	Char(4)		Selling retail unit of measure
	Selling Retail Currency	Char(3)		Selling retail currency
	Reset Clearance Id	Number(15)		Clearance reset identification
FDELE	Record Descriptor	Char(5)	FDELE	File Detail Delete Marker (1 per clearance delete)
	Line ID	Number(10)		Unique line identification
	Id	Number(15)		Clearance identifier
	Item	Char(25)		Item identifier
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker
	Line ID	Number(10)		Unique line identification
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

Design Assumptions

N/A

FutureRetailPurgeBatch Design

Table 3–30 FutureRetailPurgeBatch Details

Module Name	FutureRetailPurgeBatch.sh
Description	Purges future retail data that are past the retention period.
Functional Area	Future Retail
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	futureRetailPurgeBatch.sh <user alias>

Design Overview

This batch is a timed multi-threaded process that purges future retail data that are past the retention periods of their corresponding price events.

Scheduling Constraints

Table 3–31 FutureRetailPurgeBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc
Scheduling Considerations	This process must be executed during the batch window. As it runs, other processes must not access the future retail tables. This batch can be run ad-hoc.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	The batch uses bookmark logic to process merchandise hierarchies in a round robin fashion and running for a specific timeframe depending on the value of BATCH_TIME_LIMIT_HOURS in RPM_ BATCH_ CONTROL.

Restart/Recovery

Restart/Recovery is inherent in the design of this program, as records are deleted after processing they would not be picked up if the program is run again.

Key Tables Affected

Table 3–32 FutureRetailPurgeBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_FUTURE_RETAIL	Yes	No	No	Yes

Design Assumptions

N/A

FutureRetailRollUpBatch (Future Retail Roll Up Batch)

Table 3–33 FutureRetailRollUpBatch Details

Module Name	FutureRetailRollUpBatch.sh
Description	Attempts to roll up timelines on future retail if lower level timelines match higher levels.
Functional Area	Future Retail
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	futureRetailRollUpBatch.sh <user alias>

Design Overview

This batch attempts to roll up lower level timelines to existing higher level timelines (for example, from Item/Location to Parent/Location) by comparing two related timelines and removing the lower level timelines if the two match exactly for all records.

Scheduling Constraints

Table 3–34 FutureRetailRollUpBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc
Scheduling Considerations	This process must be executed during the batch window. As it runs, other processes must not access the future retail tables. This batch can be run ad-hoc.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	This batch is threaded by item.

Restart/Recovery

The batch uses bookmark logic to process merchandise hierarchies in a round robin fashion and running for a specific timeframe depending on the value of BATCH_ TIME_LIMIT_HOURS in RPM_ BATCH_CONTROL.

Key Tables Affected

Table 3–35 FutureRetailRollUpBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_FUTURE_RETAIL	Yes	No	Yes	No

Design Assumptions

N/A

ItemReclassBatch (Item Reclass Batch)

Table 3–36 ItemReclassBatch Details

Module Name	ItemReclassBatch.sh
Description	Updates Pricing tables when a merchandise hierarchy change is made in RMS.
Functional Area	Future Retail
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	ItemReclassBatch.sh <user alias>

Design Overview

When items are moved from one department/class/subclass to another in the merchandising system, this batch process accordingly sets the correct department/class/subclass for these items in the RPM_FUTURE_RETAIL table and the RPM_ITEM_LOC table if the item has move departments.

Scheduling Constraints

Table 3–37 ItemReclassBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc
Scheduling Considerations	Must be run during the batch window.
Pre-Processing	The RPM_ITEM_MODIFICATION table has been populated by the RMS reclassification batch process.
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 3–38 ItemReclassBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_FUTURE_RETAIL	Yes	No	Yes	No
RPM_ITEM_MODIFICATION	Yes	No	No	No

Design Assumptions

N/A

NewItemLocationBatch (New Item Location Batch Batch)

Table 3–39 NewItemLocationBatch Details

Module Name	ItemReclassBatch.sh
Description	Updates Pricing tables for new item/locations in RMS
Functional Area	Future Retail
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	NewItemLocationBatch.sh <user alias> [N / {E <error commit count>} / {R [<process ID>]}] Where The 'status' argument (N/E/R) is optional and directs the application as to what "status" to process. If it's not specified, the batch will default it to 'N'ew mode. The last argument can be optional or required depending upon the status argument as describe in the section below: Valid values for the status argument are: 'N'ew: This will process records with status of N (New) from the staging table. When the batch is run in this mode, the last argument is not needed. 'E'rror: This will process records with status of E (Error) from the staging table. When the batch is run in this mode, the batch can have the error commit count argument as an optional argument. Error commit count is optional and is used only when the status argument is 'E'. If not specified, the batch will use the logical unit of work for processing 'R'estart: When the batch is run in this mode, then the process_id argument is required. This mode will only restart the rolling up functionality that is part of location move. It will call the RPM_NEW_ITEM_LOC_SQL.ROLLUP_NIL_DATA for the threads that are not in completed status in RPM_NIL_ROLLUP_THREAD. A required valid process ID parameter will also need to be passed in as well to indicate what process ID the batch should restart.

Design Overview

The NewItemLocationBatch program ranges item locations by putting them into the future retail table. Item locations are fed to this program via the RPM_ITEM_LOC_WS table, which is populated by an RMS process.

Scheduling Constraints

Table 3–40 NewItemLocationBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc
Scheduling Considerations	Must not have more than one instance running at a time.

Table 3–40 (Cont.) NewItemLocationBatch Scheduling Constraints

Schedule Information	Description
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	The NewItemLocationBatch is a multi-step and multi-threaded batch, meaning each of the two steps (inheritance process and rollup process) has its own independent threading. The first part, which is the insert to future retail and item loc tables and inheritance process, is threaded by related item-locations where "related" means transaction items under a single parent items and locations within a zone that is part of a primary zone group. If there are price events, then it chooses a path based on batch control settings similar to the ones for a price event approval from UI, and it chooses to go to chunking or bulking based on setting and the volume of data.

Restart/Recovery

Processing Stage Rows in Error Status

This program is set up to re-process (re-attempt) rows that end up in error status. In the event that an error occurs during the processing of new status rows, the program should update the status on the stage table with E along with an error message.Once the error has been fixed, you can re-run this program with status E in an attempt to get the item/loc into RPM.

Processing Stage Rows in Restart Status

When running in Restart mode, the batch will attempt to re-process the future retail roll up functionality and to clean up item location staging tables. It will delete the records that were completely processed from the staging tables.

This mode has to be executed when there are threads/process ID that have errors or did not complete the roll-up process and clean-up of staging tables. This should be part of the business process. For example, clients can do this ad-hoc when no one is using the application. They also have to establish how they are going to retrieve process ID and threads that need reprocessing. If there won't be an established process for running NIL Batch in restart mode, the NIL thread status and staging tables data will increase and won't be cleaned up.

Key Tables Affected

Table 3–41 NewItemLocationBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_FUTURE_RETAIL	Yes	No	Yes	No
RPM_ITEM_LOC	Yes	No	No	No
RPM_STAGE_ITEM_LOC	Yes	No	Yes	No

Table 3–41 (Cont.) NewItemLocationBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_STAGE_ITEM_LOC_ CLEAN	Yes	No	No	No
RPM_NIL_ROLLUP_THREAD	Yes	No	Yes	No
RPM_NIL_BULKCCPE_ PROCESS_ID	Yes	No	No	No

Design Assumptions

N/A

NightlyBatchCleanup (Nightly Cleanup Batch)

Table 3–42 NightlyBatchCleanup Details

Module Name	NightlyBatchCleanup.sh
Description	Nightly clean up on pricing tables
Functional Area	All
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	NightlyBatchCleanupBatch.sh <user_alias> PRE/POST

Design Overview

The nightlyBatchCleanup batch program performs "clean up" logic against certain database structures.

Scheduling Constraints

Table 3–43 NightlyBatchCleanup Scheduling Constraints

Schedule Information	Description
Frequency	Nightly batch cycle
Scheduling Considerations	This batch should be run before the nightly batch window in "pre" mode and after the nightly batch window in "post" mode.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 3–44 NightlyBatchCleanup Key Tables Affected

Table	Select	Insert	Update	Delete
S9T_TEMPLATE	Yes	No	No	No
SVC_PROCESS_TRACKER	Yes	No	No	No
S9T_ERRORS	Yes	Yes	Yes	No
RPM_CORESVC_PRICE_ CHANGE_ERR	Yes	No	No	No
RPM_SVC_PRICE_CHANGE	Yes	No	Yes	No
RPM_PRICE_CHANGE	Yes	No	No	No
RPM_PRICE_CHANGE_GROUP	No	Yes	No	No

Design Assumptions

N/A

PriceChangeInductionBatch (Price Change Induction Batch)

Table 3–45 PriceChangeInductionBatch Details

Module Name	PriceChangeInductionBatch.sh
Description	Price Change bulk upload process
Functional Area	Price Change
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	PriceChangeInductionBatch .sh <user alias> <incoming-dir-path> <Template_Key> [filter_Str] 1. The first argument is the user alias name which mandatory and is mapped to an authorized user in the system. 2. The second argument is the path where the induction input files are placed. 3. Third argument is the name of the standard price change template key. This is a mandatory argument. 4. The fourth argument is an optional and when provided accepts the following values: a. XML - indicates that the batch has to look for xml files in the given incoming-dir-path and process them. b. ZIP - This is the default value when nothing is provided. The process will look for zip files containing xml files and process them sequentially.

Note: File naming standards

XML file:

The file should have a prefix of PCIND. Ex: PCIND_ABC-10.10.18.xml

The file should contain the data in the format suggested by standard price change upload template.

ZIP file:

The file should have a prefix of PCIND. Ex: PCIND_ABC.ZIP

The xml files with in the zip file should also have the prefix of PCIND

Design Overview

PriceChangeInductionBatch uploads regular price changes in bulk. For the bulk upload, price change data will be present in XML format with the data formatted in the standard price change upload template. This batch accepts the price change data present in XML format and also as zip files of xml files formatted in the standard template.

Scheduling Constraints

Table 3–46 PriceChangeInductionBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 3–47 PriceChangeInductionBatch Key Tables Affected

Table	Select	Insert	Update	Delete
S9T_TEMPLATE	Yes	No	No	No
SVC_PROCESS_TRACKER	Yes	No	No	No
S9T_ERRORS	Yes	Yes	Yes	No
RPM_CORESVC_PRICE_ CHANGE_ERR	Yes	No	No	No
RPM_SVC_PRICE_CHANGE	Yes	No	Yes	No
RPM_PRICE_CHANGE	Yes	No	No	No
RPM_PRICE_CHANGE_GROUP	No	Yes	No	No

Design Assumptions

N/A

PriceEventExecutionBatch (Price Event Execution Batch)

Table 3–48 PriceEventExecutionBatch Details

Module Name	PriceEventExecution.sh
Description	Starts events that need to be executed on a given date.
Functional Area	Price Change
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	`PriceEventExecutionBatch.sh <user_alias> [restartInd Y|N]` Where the last argument of the PriceEventExecutionBatch indicates if the run should start over (use a value of N) or pick up where the previous run left off (use a value of Y).

Design Overview

The price event execution batch process performs the necessary work to start (regular price change and clearance price change) and end (reset) clearance pricing events.

The batch programs process regular price change and clearance price change events that are scheduled for the run date. Restartability features allow events missed in past runs of the batch to be picked up in later runs. When posting information in the ITEM_LOC and PRICE_HIST table, the batch process respects the active dates of their associated price events.

Clearances

- Clearance markdowns that are scheduled to take place are executed. These include all clearances whose effective dates are <= VDATE+1.
- Clearances that are scheduled to be completed (reset) are completed.

Regular price changes

- Regular price changes that are scheduled to take place are executed. These include all price changes whose effective dates are <= VDATE+1.

Scheduling Constraints

Table 3–49 PriceEventExecutionBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring
Scheduling Considerations	Salstage (RMS) should run before Price Event Execution. Price Event Execution should run before the Storadd (RMS) batch.

Table 3–49 (Cont.) PriceEventExecutionBatch Scheduling Constraints

Schedule Information	Description
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

The program is restartable and will pick up any events remaining to be processed in a given run.

Key Tables Affected

Table 3–50 PriceEventExecutionBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_PRICE_CHANGE	Yes	No	Yes	No
RPM_CLEARANCE	Yes	No	Yes	No
ITEM_LOC	Yes	No	Yes	No
PRICE_HIST	Yes	Yes	No	No
TRAN_DATA	Yes	Yes	No	No

Design Assumptions

N/A

Purge Batch (PurgeBatch)

Here are the steps in the purge process:

- Delete items and item/locations which were already deleted in the merchandising system from the pricing system.
- Delete price changes which are in worksheet, rejected, or submitted status with an effective date beyond the reject hold days.
- Remove price changes with effective dates beyond the price change history months retention setting. This is completed by dropping partitions which meet the date criteria.
- Delete clearances which are in worksheet, rejected, or submitted status with an effective date beyond the reject hold days.
- Remove clearances with effective dates beyond the clearance history months retention setting. This is completed by dropping partitions which meet the date criteria.
- Delete all but the most recent zone future retail entry with an effective date before vdate.
- Delete price change induction data for successful upload processes and those with an action date beyond the process retention days.
- Delete clearance induction data for successful upload processes and those with an action date beyond the process retention days.

- Delete published payload data from price changes, clearances, and promotions that is older than 7 days.
- Truncate all the Bulk CC processing tables.
- Delete conflict check error results for price changes and clearances which no longer exist.
- Truncate all UI, item creation, and item/location ranging workspace tables.

System Options

System options used for purge configuration:

- RPM_PURGE_CONFIG_OPTIONS.PRICE_EVENTS_REJECT_HOLD_DAYS
- RPM_PURGE_CONFIG_OPTIONS.PRICE_CHANGE_HIST_MONTHS
- RPM_PURGE_CONFIG_OPTIONS.CLEARANCE_HIST_MONTHS
- SYSTEM_OPTIONS.PROC_DATA_RETENTION_DAYS

Usage

PurgeBatch.sh <user_alias>PurgeBatch.sh <user_alias> <export_purge ALL/BDI/FLAT_FILE>

Where ALL is used if the customer is using both BDI integration and flat file integration, BDI for BDI only, or FLAT_FILE for flat file only.

Scheduling Constraints

Table 3–51 PurgeBatch Scheduling Constraints

Schedule Information	Description
Frequency	Nightly
Scheduling Considerations	not applicable
Pre-Processing	not applicable
Post-Processing	not applicable
Threading Scheme	not applicable

Restart/Recovery

N/A

Key Tables Affected

Table 3–52 PurgeGTTCaptureBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_ZONE_FUTURE_RETAIL	Yes	No	No	Yes
RPM_ITEM_LOC	Yes	No	No	Yes
RPM_ZONE_LOCATION	Yes	No	No	Yes
RPM_ITEM_ZONE_PRICE	Yes	No	No	Yes
RPM_PRICE_CHANGE	Yes	No	No	Yes

Table 3–52 (Cont.) PurgeGTTCaptureBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_CLEARANCE	Yes	No	No	Yes
RPM_STAGE_DELETED_ITEM_MASTER	Yes	No	No	Yes
RPM_STAGE_DELETED_ITEM_LOC	Yes	No	No	Yes
RPM_PRICE_CHANGE	Yes	No	No	Yes
RPM_PRICE_CHANGE_GROUP	Yes	No	No	Yes
RPM_CLEARANCE	Yes	No	No	Yes
RPM_CLEARANCE_GROUP	Yes	No	No	Yes
RPM_ZONE_FUTURE_RETAIL	Yes	No	No	Yes
SVC_PROCESS_TRACKER	Yes	No	No	Yes
S9T_FOLDER	Yes	No	No	Yes
S9T_ERRORS	Yes	No	No	Yes
RPM_SVC_PRICE_CHANGE	Yes	No	No	Yes
RPM_CORESVC_PRICE_CHANGE_ERR	Yes	No	No	Yes
SVC_PROCESS_TRACKER	Yes	No	No	Yes
S9T_FOLDER	Yes	No	No	Yes
S9T_ERRORS	Yes	No	No	Yes
RPM_SVC_CLEARANCE	Yes	No	No	Yes
RPM_CORESVC_CLEARANCE_ERR	Yes	No	No	Yes
RPM_PROMO_OFFER_CANCEL_PAYLOAD	Yes	No	No	Yes
RPM_PROM_OFR_CNCL_LOC_PAYLOAD	Yes	No	No	Yes
RPM_PROM_OFR_CNCL_ITEM_PAYLOAD	Yes	No	No	Yes
RPM_PROMO_OFFER_LOC_PAYLOAD	Yes	No	No	Yes
RPM_PROMO_OFR_RWD_MRCH_PAYLOAD	Yes	No	No	Yes
RPM_PROMO_OFR_CND_MRCH_PAYLOAD	Yes	No	No	Yes
RPM_PROMO_OFFER_PAYLOAD	Yes	No	No	Yes
RPM_PRICE_CHG_PAYLOAD	Yes	No	No	Yes
RPM_CLEARANCE_PAYLOAD	Yes	No	No	Yes
RPM_PRICE_EVENT_PAYLOAD	Yes	No	No	Yes
RPM_BULK_CC_PE_ITEM	Yes	No	No	Yes
RPM_BULK_CC_PE_LOCATION	Yes	No	No	Yes
RPM_BULK_CC_PE_CHUNK	Yes	No	No	Yes

Table 3–52 (Cont.) PurgeGTTCaptureBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_BULK_CC_PE_THREAD	Yes	No	No	Yes
RPM_BULK_CC_PE_SEQUENCE	Yes	No	No	Yes
RPM_BULK_CC_PE	Yes	No	No	Yes
RPM_PE_CC_LOCK	Yes	No	No	Yes
RPM_CONFLICT_CHECK_ RESULT	Yes	No	No	Yes
RPM_CON_CHECK_ERR	Yes	No	No	Yes
RPM_CON_CHECK_ERR_ DETAIL	Yes	No	No	Yes
RPM_PC_MAINT_LOC_WS	Yes	No	No	Yes
RPM_PC_MAINT_ITEM_WS	Yes	No	No	Yes
RPM_PC_MAINT_WS	Yes	No	No	Yes
RPM_PC_GROUP_SEARCH_WS	Yes	No	No	Yes
RPM_CLR_MAINT_LOC_WS	Yes	No	No	Yes
RPM_CLR_MAINT_ITEM_WS	Yes	No	No	Yes
RPM_CLR_MAINT_WS	Yes	No	No	Yes
RPM_CLR_GROUP_SEARCH_ WS	Yes	No	No	Yes
RPM_OFFER_ZONE_LOC_WS	Yes	No	No	Yes
RPM_OFFER_CON_RWD_ MERCH_WS	Yes	No	No	Yes
RPM_OFFER_REWARD_WS	Yes	No	No	Yes
RPM_OFFER_COND_WS	Yes	No	No	Yes
RPM_OFFER_WS	Yes	No	No	Yes
RPM_PROMO_WS	Yes	No	No	Yes
RPM_PROMO_OFFER_ SEARCH_WS	Yes	No	No	Yes
RPM_PROMO_CANCEL_ MERCH_WS	Yes	No	No	Yes
RPM_PROMO_CANCEL_ ZONE_NODE_WS	Yes	No	No	Yes
RPM_PROMO_OFFER_PUB_WS	Yes	No	No	Yes
RPM_PE_CREATE_ITEM_WS	Yes	No	No	Yes
RPM_PE_CREATE_LOC_WS	Yes	No	No	Yes
RPM_PE_CREATE_WS	Yes	No	No	Yes
RPM_PE_CREATE_SUMMARY_ WS	Yes	No	No	Yes
RPM_OI_PC_PEND_APPRV_ EOW	Yes	No	No	Yes
RPM_OI_PC_PEND_APPRV_ DAY	Yes	No	No	Yes

Table 3–52 (Cont.) PurgeGTTCaptureBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_OI_CLR_PEND_APPRV_EOW	Yes	No	No	Yes
RPM_OI_CLR_PEND_APPRV_DAY	Yes	No	No	Yes
RPM_OI_UPCOMING_OFFER_WS	Yes	No	No	Yes
RPM_ROWID_TEMP	Yes	No	No	Yes
RPM_STAGE_ITEM_LOC_RETAIL_TEMP	Yes	No	No	Yes

PurgeGTTCaptureBatch (Purge GTT Capture Batch)

Table 3–53 PurgeGTTCaptureBatch Details

Module Name	PurgeGttCaptureBatch.sh
Description	Truncates data from the GTT capture related tables.
Functional Area	Various
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	`PurgeGttCaptureBatch.sh <user_alias>`

Design Overview

This batch truncates data from the GTT capture related tables.

Scheduling Constraints

Table 3–54 PurgeGTTCaptureBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc
Scheduling Considerations	Should be run during batch window.
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	N/A

Restart/Recovery

N/A

Key Tables Affected

Table 3–55 PurgeGTTCaptureBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_RFR_GTT_DATA_ CAPTURE	Yes	No	No	No
RPM_RPILE_GTT_DATA_ CAPTURE	Yes	No	No	No
RPM_CSPFR_GTT_DATA_ CAPTURE	Yes	Yes	Yes	No
RPM_CLR_GTT_DATA_ CAPTURE	Yes	No	No	No
RPM_FRILE_GTT_DATA_ CAPTURE	Yes	No	Yes	No

Design Assumptions

N/A

RegularPriceChangePublishBatch (Regular Price Change Publish Batch)

Table 3–56 RegularPriceChangePublishBatch Details

Module Name	RegularPriceChangePublishBatch.sh
Description	Price Change events are exported for integration to other systems.
Functional Area	Price Changes
Module Type	Business Processing
Module Technology	Java
Catalog ID	
Runtime Parameters	`RegularPriceChangePublishBatch.sh <user_alias> <outgoing-dir-path>`

Design Overview

The RegularPriceChangePublishBatch program formats and stages output of regular price change price events.

The corresponding regularPriceChangePublishExport shell script produces a pipe ("|") delimited flat-file export based on the output of the RegularPriceChangePublishBatch.

The batch looks for price events in the RPM_PRICE_EVENT_PAYLOAD table with a RIB_FAMILY of "REGPRCCHG" and distributes those events to multiple threads based on the settings in the RPM_BATCH_CONTROL table. Each thread reads in its set of regular price change events from tables RPM_PRICE_EVENT_PAYLOAD and RPM_PRICE_CHG_PAYLOAD and generates output in RPM_PRICE_PUBLISH_ DATA.

A flat-file per location based on the data from payload table that need to be published/processed will be created. The naming convention for the flat file will be (REGPC_<timestamp> _<location>_<loc_type>.dat), where <timestamp> is the

current system time stamp, <location> is the location ID and <loc_type> is the type of the location where 'S' is for Store and 'W' is for Warehouse.

Scheduling Constraints

Table 3–57 RegularPriceChangePublishBatch Scheduling Constraints

Schedule Information	Description
Frequency	Ad hoc, Recurring
Scheduling Considerations	N/A
Pre-Processing	N/A
Post-Processing	N/A
Threading Scheme	The RegularPriceChangePublishBatch program is threaded, using RPM_BATCH_CONTROL. The LUW is a single price change event.

Restart/Recovery

N/A

Key Tables Affected

Table 3–58 RegularPriceChangePublishBatch Key Tables Affected

Table	Select	Insert	Update	Delete
RPM_PRICE_EVENT_PAYLOAD	Yes	No	No	No
RPM_PRICE_CHG_PAYLOAD	Yes	No	No	No

Output Files

FHEAD (required): File identification, one line per file.

FDETL (optional): Price Change Event (Create or Modify).

FDELE (optional): Price Change Event (Delete).

FTAIL (required): End of file marker, one line per file.

Note: File naming standards

The naming convention for the flat file will be (REGPC_<timestamp>_<location>_<loc_type>.dat), where <timestamp> is the current system time stamp, <location> is the location ID and <loc_type> is the type of the location where 'S' is for Store and 'W' is for Warehouse. The zip file naming convetion will be (REGPC_<timestamp>.zip).

Output File Layout

Table 3–59 Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FHEAD	Record Descriptor	Char(5)	FHEAD	File head marker
	Line ID	Number(10)	1	Unique line identifier
	File Type	Char(5)	REGPC	Regular Price Changes
	Export timestamp	Timestamp		System clock timestamp (YYYYMMDDHHMISS)
	Location	Number(10)		Location identifier
	Location Type	Char(1)		S = Store, W = Warehouse
FDETL	Record Descriptor	Char(5)	FDETL	File Detail Marker (1 per price change create or modify)
	Line ID	Number(10)		Unique line identifier
	Event Type	Char(3)		CRE = Create, MOD = Modify
	Id	Number(15)		Price Change identifier
	Item	Char(25)		Item identifier
	Effective Date	Date		Effective Date of price change (YYYYMMDDHH24MISS)
	Selling Unit Change Ind	Number(1)		Did selling unit retail change with this price event (0 = no change, 1 = changed)
	Selling Retail	Number(20,4)		Selling retail with price change applied
	Selling Retail UOM	Char(4)		Selling retail unit of measure
	Selling Retail Currency	Char(3)		Selling retail currency
	Multi-Unit Change Ind	Number(1)		Did multi-unit retail change with this price event (0 = no change, 1 = changed)
	Multi-Units	Number(12,4)		Number of multi-units
	Multi-Unit Retail	Number(20,4)		Multi-Unit Retails
	Multi-Unit UOM	Char(4)		Multi-Unit Retail Unit Of Measure
	Multi-Unit Currency	Char(3)		Multi-Unit Retail Currency
FDELE	Record Descriptor	Char(5)	FDELE	File Detail Delete Marker (1 per price change delete)
	Line ID	Number(10)		Unique line identifier
	Id	Number(15)		Price Change identifier
	Item	Char(25)		Item identifier

Table 3–59 (Cont.) Output File Layout

Record Name	Field Name	Field Type	Default Value	Description
FTAIL	Record Descriptor	Char(5)	FTAIL	File tail marker
	Line ID	Number(10)		Unique line identifier
	Number of lines	Number(10)		Number of lines in file not counting FHEAD and FTAIL

Design Assumptions

N/A

4

Induction Services

This chapter describes the induction services. The primary role of these services is to create, modify, and delete price changes and clearances.

Price Change Induction Service

This service creates, modifies, or deletes price changes by calling the RPM_PRICE_CHANGE_INDUCTION_SQL package to load input data into staging tables and then calls the core RPM_CORESVC_PRICE_CHANGE_SQL package to validate and insert data to the RPM price change tables.

Service Type

POST

REST URL

/priceChange/induction

Input Parameters

Table 4–1 PriceChangeInductionRDO Input Parameters

Parameter Name	Data Type
action	String
newGroupBatch	Long
priceChangeGroup	Long
priceChangeGroupDesc	String
priceChange	Long
item	String
diff	String
locationType	String
location	Long
effectiveDate	Localdate
updatedEffectiveDate	Localdate
changeType	String
changeValue	BigDecimal

Table 4–1 (Cont.) PriceChangeInductionRDO Input Parameters

Parameter Name	Data Type
sellingUom	String
roundingRule	String
reason	String
status	String
ignoreConstraints	String

JSON Structure

```
[
  {
    "action": null,
    "newGroupBatch": null,
    "priceChangeGroup": null,
    "priceChangeGroupDesc": null,
    "priceChange": null,
    "item": null,
    "diff": null,
    "locationType": null,
    "location": null,
    "effectiveDate": null,
    "updatedEffectiveDate": null,
    "changeType": null,
    "changeValue": null,
    "sellingUom": null,
    "roundingRule": null,
    "reason": null,
    "status": null,
    "ignoreConstraints": null
  }
]
```

Output

The output will contain the status of the request including validation errors, if any.

Table 4–2 PriceChangeStatusRDO Ouput Parameters

Parameter Name	Data Type
statusMsg	String
failPcTable	List<PriceChangeFailRDO>

Table 4–3 PriceChangeFailRDO Ouput Parameters

Parameter Name	Data Type
priceChange	Long
item	String
diff	String
locationType	String
location	Long

Table 4–3 (Cont.) PriceChangeFailRDO Ouput Parameters

Parameter Name	Data Type
effectiveDate	LocalDate
errorMsg	String

JSON Structure

```
[
  {
    "statusMsg": null,
    "failPcTable": [
      {
        "priceChange": null,
        "item": null,
        "diff": null,
        "locationType": null,
        "location": null,
        "effectiveDate": null,
        "errorMsg": null
      }
    ],
    "links": [],
    "hyperMediaContent": {
      "linkRDO": []
    }
  }
]
```

Table Impact

Table 4–4 Price Change induction Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
RPM_SVC_PRICE_CHANGE	Yes	Yes	Yes	No
RPM_CORESVC_PRICE_ CHANGE_ERR	Yes	Yes	No	No
RPM_PRICE_CHANGE	Yes	Yes	Yes	Yes

Clearance Induction Service

This section describes the Clearance Induction service. This service creates, modifies, or deletes clearances by calling the RPM_CLEARANCE_INDUCTION_SQL package to load input data into staging tables and then calls the core RPM_CORESVC_ CLEARANCE_SQL package to validate and insert data to the RPM clearance tables.

Service Type

POST

REST URL

/clearance/induction

Input Parameters

Table 4–5 ClearanceInductionRDO Input Parameters

Parameter Name	Data Type
action	String
newGroupBatch	Long
clearanceGroup	Long
clearanceGroupDesc	String
clearance	Long
markdown	String
item	String
diff	String
locationType	String
location	Long
effectiveDate	Localdate
updatedEffectiveDate	Localdate
changeType	String
changeValue	BigDecimal
roundingRule	String
reason	String
status	String

JSON Structure

```
[
  {
    "action": null,
    "newGroupBatch": null,
    "clearanceGroup": null,
    "clearanceGroupDesc": null,
    "clearance": null,
    "markdown": null,
    "item": null,
    "diff": null,
    "locationType": null,
    "location": null,
    "effectiveDate": null,
    "updatedEffectiveDate": null,
    "changeType": null,
    "changeValue": null,
    "roundingRule": null,
    "reason": null,
    "status": null
  }
]
```

Output

The output will contain the status of the request including validation errors, if any.

Table 4–6 ClearanceStatusRDO Ouput Parameters

Parameter Name	Data Type
statusMsg	String
failClrTable	List<ClearanceFailRDO>

Table 4–7 ClearanceFailRDO Ouput Parameters

Parameter Name	Data Type
clearance	Long
item	String
diff	String
locationType	String
location	Long
effectiveDate	LocalDate
errorMsg	String

JSON Structure

```
[
  {
    "statusMsg": null,
    "failClrTable": [
      {
        "clearance": null,
        "item": null,
        "diff": null,
        "locationType": null,
        "location": null,
        "effectiveDate": null,
        "errorMsg": null
      }
    ],
    "links": [],
    "hyperMediaContent": {
      "linkRDO": []
    }
  }
]
```

Table Impact

Table 4–8 Clearance Induction Table Impact

TABLE	SELECT	INSERT	UPDATE	DELETE
SVC_PROCESS_TRACKER	Yes	Yes	Yes	No
RPM_SVC_CLEARANCE	Yes	Yes	Yes	No
RPM_CORESVC_CLEARANCE_ERR	Yes	Yes	No	No
RPM_CLEARANCE	Yes	Yes	Yes	Yes

5

Backend System Administration and Configuration

This chapter of the operations guide is intended for administrators who provide support and monitor the running system.

Supported Environments

See the *Oracle Retail Price Management Installation Guide* for information about requirements for the following:

- RDBMS operating system
- RDBMS version
- Middle tier server operating system
- Middle tier
- Compiler

Exception Handling

The two primary types of exceptions within the RPM system are the following:

- System exceptions

  For example, server connection and/or database issues are system exceptions. System exceptions can bring the system to a halt. For example, the connection to the server is lost.

- Business exceptions

  This exception indicates that a business rule has been violated. Most exceptions that arise in the system are business exceptions. For example, a user tries to approve a price change that causes a negative retail.

Logging Configuration

Logging within RPCS utilizes the ADF built-in logging framework to log system messages and exceptions. This framework is embedded in the application code to allow for configurable logging to suit the needs of the retailer.

Please note that batch client programs log the messages, errors to a log file configured in batch_logging.properties. Server logging is done using standard WebLogic logging infra structure.

ADF Logging

This is a wrapper class of java logger class. It adds ADF conveneience methods. All other java logger methods as well are available for user. The following are the different logging levels possible.

- SEVERE (highest value)
- WARNING
- INFO
- CONFIG
- FINE
- FINER
- FINEST (lowest value)

Note: In a production environment, the logging setting should be set to Severe or Warning, so that system performance is not adversely impacted.

Batch Client Logging

The pricing batch client java programs write error messages, warnings to a log file configured in batch_logging.properties. The logging mechanism is based on FileHandler java API.

By default, the log file is configured to be created in the logs folder under user home directory (%h) with the name batch_log appended with a random number (%u). See below batch_logging properties file more details.

Batch_logging Properties

The batch_logging.properties file holds all of the information relevant to logging for batch clients.

Table 5–1

Parameter	Description
Handlers	A comma-delimited list of handler class names that are added to the root Logger. The default handlers are java.util.logging.FileHandler andjava.util.logging.ConsoleHandler (with a default level of INFO).
.level	Sets the log level for all FileHandler instances. The default log level is INFO.
java.util.logging.FileHandler.pattern	The log file name pattern. The default is %h/../logs/batch_log%u.log which means that the file is named batch_log%u.log where:. %h the value of the "user.home" system property %u is a unique number to resolve conflicts between simultaneous Java processes

Table 5–1 (Cont.)

Parameter	Description
java.util.logging.FileHandler.limit	The maximum size of the file, in bytes. If this is 0, there is no limit. The default is 1000000 (which is 1 MB). Logs larger than 1MB roll over to the next log file.
java.util.logging.FileHandler.count	The number of log files to use in the log file rotation. The default is 365 (which produces a maximum of 365 log files).
java.util.logging.FileHandler.level	Sets the log level for all FileHandler instances. The default log level is FINEST.
java.util.logging.ConsoleHandler.level	Sets the default log level for all ConsoleHandler instances. The default log level is FINEST..
java.util.logging.FileHandler.append	Specifies whether the FileHandler should append onto any existing files (defaults to true)

Configurable GTTCapture

The conflict checking engine within RPM utilizes Global Temporary Tables (GTT) extensively which allow for a performance gain, but means that transactional data is lost when the process completes. When attempting to troubleshoot issues within the conflict checking engine around GTT data, this leads to difficulty researching and recreating issues.

A configuration within RPM allows for capturing this GTT data while processing through the conflict checking engine in an autonomous fashion so that the data is available for review after the process has completed. Data can be captured from the following set of tables:

- RPM_FUTURE_RETAIL_GTT
- RPM_PROMO_ITEM_LOC_EXPL_GTT
- RPM_CUST_SEGMENT_PROMO_FR_GTT
- RPM_CLEARANCE_GTT
- RPM_FR_ITEM_LOC_EXPL_GTT

The system is designed to capture data from any of these GTTs based on configuration. Data can be captured from one or more of these tables during conflict checking and can be captured at a configurable start point and optionally beyond the starting point. There are five options for starting points when capturing GTT data:

- GTT Initial Population
- Merge Price Event into Timelines
- Roll Forward
- Payload Population
- Future Retail Purge

The system will also allow for specifying if GTT data should be captured for a specific user in the system or for any user. When specifying a user id to capture data for, the user id needs to match with the user defined within LDAP and should have matching case between LDAP and the GTT capture configuration.

All configuration is handled via the RPM_CONFIG_GTT_CAPTURE table by direct table updates. It is possible to set up all the necessary configurations (starting point, specific user, capture data beyond start point and what tables to capture data from) and disable the capturing of this data all together by setting the ENABLE_GTT_CAPTURE field to 'N'. Once the GTT capture configurations are established and enabled on the RPM_CONFIG_GTT_CAPTURE table, nothing more needs to be done other than to process a price event through conflict checking.

When the system does capture data from the GTT tables, it will always capture all data on the specified tables at the "starting point" and then only capture updated or newly created data for each statement beyond that point when data is being captured beyond the starting point. In such a scenario, the evolution of a record will be easily available for viewing and troubleshooting efforts with the impact of every statement being identified easily.

A batch process (PurgeGttCaptureBatch.sh) will purge all data captured from the GTT tables to allow for only pertinent data to be in place at any given time. This purge process does not have to run prior to capturing GTT data in conflict checking, however it is expected that capturing this data will produce a large volume of data in many scenario. By purging this data before running the conflict checking process again for new data to be captured, it will be easier to examine the data.