Retek® Merchandising System™ 9.0.13

Operations Guide Addendum

Retek

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

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@retek.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

Pre/Post Functionality for Multi-Threadable Programs [prepost]....... 1

Design Overview 1

Scheduling Constraints 2

Restart Recovery 2

Program Flow 2

Shared Modules 2

Function Level Description 3

I/O Specification 6

Technical Issues 6

Store Add [storeadd]....... 7

Design Overview 7

Scheduling Constraints 7

Restart/Recovery 7

Program Flow 7

Function Level Description 8

I/O Specification 10

Technical Issues 10

Pre/Post Functionality for Multi-Threadable Programs [prepost]

Design Overview

The Pre/Post module facilitates multi-threading by allowing general system administration functions (such as table deletions or mass updates) to be completed after all threads of a particular program have been processed. A brief description of all pre- or post-processing functions included in this program can be found in the Function-Level Description section.

This program will take three parameters: username/password to log on to Oracle, a program before or after which this script must run and an indicator telling whether the script is a pre or post function. It will act as a shell script for running all pre-program and post-program updates and purges (the logic was removed from the programs themselves to enable multi-threading & restart/recovery).

For example, to run the pre-program script for the ccext program, the following should be entered on the command line:

```
prepost    user/password    ccext    pre
```

Tables affected:

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
COST_SUSP_SUP_HEAD		No	No	Yes	No
MOD_ORDER_ITEM_HTS		No	NO	No	Yes
ON_ORDER_TEMP		No	No	No	Yes
ORDHEAD		No	No	Yes	No
PERIOD		Yes	No	No	No
POS_MODS		No	No	No	Yes
WH_STORE_ASSIGN		No	No	No	Yes
SYSTEM_OPTIONS	No	Yes	No	No	No
SYSTEM_VARIABLES	No	Yes	No	Yes	No
STORE	Yes	Yes	No	Yes	No
STORE_ADD	Yes	Yes	No	No	Yes
PRICE_BATCH_TRAN	No	No	No	No	Yes
SUP_DATA	Yes	No	No	No	Yes

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
DEAL_CALC_QUEUE_TEMP	No	Yes	Yes	No	No
POS_COUPON_HEAD	Yes	No	No	Yes	No
POS_PROD_REST_HEAD	Yes	No	No	Yes	No
POS_MONEY_ORD_HEAD	Yes	No	No	Yes	No
POS_SUP_DAY_CRITERIA	Yes	No	No	Yes	No
POS_PAYINOUT_HEAD	Yes	No	No	Yes	No
POS_BUTTON_HEAD	Yes	No	No	Yes	No
POS_TENDER_TYPE_HEAD	Yes	No	No	Yes	No

Scheduling Constraints

Processing Cycle: PHASE ALL (daily)

Scheduling Diagram: See scheduling flow for description of all pre-post requirements in the daily run.

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: N/A (single threaded)

Restart Recovery

N/A

Program Flow

N/A

Shared Modules

N/A

Function Level Description

salweek_post():

- Updates the last end-of-week date on the SYSTEM_VARIABLES table to today's date after all weekly stock ledger data has been processed.

salmth_post():

- Updates the following SYSTEM_VARIABLES columns to reflect the current date's values after all monthly stock ledger data has been processed:
  - last_eom_half_no
  - last_eom_month_no
  - last_eom_date
  - next_eom_date
  - last_eom_start_half
  - last_eom_end_half
  - last_eom_start_month
  - last_eom_mid_month
  - last_eom_next_half_no
  - last_eom_day
  - last_eom_week
  - last_eom_month
  - last_eom_year
  - last_eom_week_in_half

rpl_pre():

- Truncates the following tables before replenishment extracts are performed:
  - ORD_TEMP
  - ORD_SUGGF
  - ORD_SUGG

pcovrl_pre():

- Truncates the PRICE_CONFLICT table before new conflicts are found and written.

rplatupd_pre():

- Truncates the MC_REJECTIONS table so that it is free to hold new mass change rejections.

rplatupd_post():

- Truncates the holding tables REPL_ATTR_UPDATE_ITEM and REPL_ATTR_UPDATE_LOC after their records have been processed.

pctranex_post():

- Deletes from the table PRICE_BATCH_TRAN after the price changes it held have been processed.

supmth_post():

- Truncates the table SUP_DATA after all daily supplier data records have been rolled up to month level.

sccext_post():

- Updates all processed supplier cost change record statuses to 'Extracted'.

rplbld_pre():

  - Updates replenishment order origins from 'Current System Generated' to 'Past System Generated' before processing. Enables the following triggers:
    - RMS_TABLE_OHE_AIUR
    - RMS_COL_OHE_STATUS_AUR
    - RMS_TABLE_ALH_AIDR
    - RMS_TABLE_OSK_AIUR
    - RMS_COL_ALD_QA_AUR
    - RMS_TABLE_ALD_AIUDR
    - RMS_TABLE_OLO_AUR
    - RMS_COL_OLO_QTYORDERED_AUR

rplprg_post():

- Disables the following triggers:
    - RMS_TABLE_OHE_AIUR
    - RMS_COL_OHE_STATUS_AUR
    - RMS_TABLE_ALH_AIDR
    - RMS_TABLE_OSK_AIUR
    - RMS_COL_ALD_QA_AUR
    - RMS_TABLE_ALD_AIUDR
    - RMS_TABLE_OLO_AUR
    - RMS_COL_OLO_QTYORDERED_AUR

hstbld_post():

- Truncates the holding table MASK_REBUILD after building history records.

posdnld_post():

- Clears the POS_MODS table after all records have been downloaded to the POS.

poscdnld_post():

- Clears the config_status and loc_grp_status in POS_LOC_GRP. Sets all values of extract_req_ind to ‘N’.
- Clears the status column in POS_MERCH_CRITERIA.
- Sets the status_ind column in POS_STORE to ‘N’.

prmext_post():

- Updates the promotion item status on the PROMSKU table, setting it to NULL if the promotion item has been extracted successfully or to ‘Delete Processed’ if the item was marked for deletion before extraction.

redidlprd_post():

Note: This function should be run after the edidlprd program.

It needs to be run from the command line as follows:

```
prepost  username/password  edidlprd  post
```

- Deletes old records from the EDI_DAILY_SALES table after they have been processed.

fcstrbld_post():

- Truncates the holding table FORECAST_REBUILD after all records have been processed.

vrplbld_post():

- Truncates the EDI_ORD_TEMP table after all replenishment orders have been build from the data held there.

fsadnld_post():

- Updates the load_sales_ind to ‘N’ for all records on the appropriate domain table – domain_dept, domain_class, or domain_subclass, where system_options.domain_level = ‘D’, ‘C’, or ‘S’, respectively.

whstrasg_post():

- deletes all warehouse store assignment records from the warehouse store assignment table if the assignment date (wh_store_assign.assign_date) is less than or equal to the current date (period.vdate) minus the warehouse store assignment history days (system_options.wh_store_assign_hist_days).

cntrordb_post()

- Set the last_cont_order_date on system_variables to vdate.

fifgldn1_post()

- If Oracle Financials is being used, delete everything from the fif_receiving table and repopulate it from the if_tran_data table

sub_items_delete_post():

- Delete items that are not on repl_item_loc from sub_items_detail and sub_items_head.

policy_enable():

- enables or disables policies

truncate_user_sec_table()

- truncates the user security tables and prepares for the user product security rebuild, user location security rebuild, or user zone security rebuild.

update_security_indicator():

- resets the update_sec_xxx_ind to 'N' where xxx represents prd, loc or zon if it is a user product security rebuild, user location security rebuild or user zone security rebuild, respectively

tifposdn_post()

- truncates the tif_explode table

dealcalc_post():

- Truncates the deal_sku_temp table.

htsupld_pre()

- Truncates the mod_order_item_hts table so that reports will be correct and not include data from previous runs of htsupld.

onordext_pre()

- Truncates the on_order_temp table so onordext will not include previous data.

likestore_post()

- Copies the item replenishment information to the new store. Purges the store_add table. Updates the store_open_date and store_close_date fields in the store table to the original values specified in the Store Maintenance form.

I/O Specification

N/A

Technical Issues

N/A

Store Add [storeadd]

Design Overview

This program will add all information necessary for a new store to function properly. When a store is added to the system, the store will be accessible in the system only after storeadd.pc, likestore.pc and prepost.pc are run.

The batch program loops through each record on the store_add table.

It also supports the replenishment system in RMS 9.0.

Scheduling Constraints

Processing Cycle:	Daily, Ad Hoc Phase
Scheduling Diagram:	N/A
Pre-Processing:	pcext.pc pcdnld.pc
Post-Processing:	likestore.pc, prepost.pc and slocrbld.pc
Threading Scheme:	Table based processing, don't use multithreading.

Restart/Recovery

Select ALL FIELDS from store_add.

After successfully processing a record in store_add table that does not utilize the likestore functionality, the storeadd.pc batch program will immediately delete that record in the store_add table.

For records in the store_add table that utilize the likestore functionality, the storeadd.pc batch program will not delete the record immediately in the store_add table. Instead, it will set the newly inserted record in the store table to have an open and close date far into the future. This will effectively mark that record as being already processed by storeadd.pc batch program. This also prevents the new store from being used in the system until the entire string of store addition programs is run. The prepost.pc batch program which runs after the storeadd.pc and likestore.pc will eventually delete the records in the store_add table and will set the open store date back to the correct date.

The deletion of the records in the store_add table and the changing of the open store dates provide implicit restart recovery for storeadd.pc

Program Flow

N/A

Function Level Description

init()

Declare restart variables

Get system variables (ELC indicator and pricing rule)

process()

```
Loop through store_add table
Set “new” variable indicators
Insert into store table
Call Insert_Pricing_Zone
If elc_ind = ‘Y’
  Call Insert_Cost_Zones
end if;
If copy_dlvry_ind = ‘Y’
  Call Copy_Dlvry_Sched
End if;
If copy_close_ind = ‘Y’
  Call Copy_Close_Sched
End if;
If copy_rws_info_ind = ‘Y’
  Call copy_rws_info
End if;
Call Insert_Stock_Loc_Traits
If likestore_ind == ‘N’
  Delete from store_add
End if;
```

Insert_Pricing_Zone()

This function inserts records into pricing zone tables as is appropriate to the store being created:

```
insert corporate pricing zone information
insert store pricing zone information
call Item_Zone_Price
if new_price_zone_ind = ‘N’
  insert zone info for existing currency
else
```

```
  insert new zone info
  call Item_Zone_Price (to add appropriate record for the new zone)
```

Insert_Cost_Zones()

This function inserts records into cost zone table as is appropriate to the store being created:

```
insert corporate cost zone information
insert store cost zone information
if new_cost_zone_ind = 'N'
  insert cost zone detail records
else
  insert new zone
```

Item_Zone_Price()

This function inserts records into the item_zone_price table for a new pricing zone after it's been created. This function utilizes array processing.

Copy_Store_Items()

This function calls the like_store_execute_sql.copy_store_items package function, which copies all item/store records from the like_store and inserts them for the new store.

Copy_Repl_Info()

This function copies all replenishment information for items from the selected like_store and copies them into replenishment tables for the new store.

Copy_Dlvry_Sched()

This function copies all the location delivery schedules from the selected like_store and copies them into the source_dlvry_sched, source_dlvery_sched_days, and source_dlvry_sched_exc tables for the new store.

Copy_Close_Sched()

This function copies all the location closed information from the selected like_store which the close_date are greater or equal to current and copies them into location_closed and company_closed_excep tables for the new store.

Insert_Stock_Loc_Traits()

This function calls the stkledgr_sql.stock_ledger_insert and loc_traits_sql.new_org_hier package functions, which insert records into the stock ledger and hierarchy tables.

Insert_Pos_Store()

This function inserts data to the pos_store table**.**

Copy_Rws_Info()

This function inserts records to the lif_location_detail table.

Size_Izp_Arrays()

This function allocates memory for array processing.

final()

This function stops restart recovery.

I/O Specification

N/A

Technical Issues

N/A