Retek™

Retek Merchandising System 9.0.4

Addendum to Operations Guide

rms-904-og-addendum

Retek Merchandising System™

Copyright Notice

Trademarks

Policy on Retek End User Documentation

Customer Support

Customer Support hours:

8 AM to 5 PM Central Standard Time (GMT-6), Monday through Friday, excluding Retek company holidays (in 2001: Jan. 1, May 28, July 4, Sept. 3, Nov. 22, Nov. 23, Dec. 24, and Dec. 25).

Customer Support emergency hours:

24 hours a day, 7 days a week.

Contact Method	Contact Information
Phone	US & Canada: 1-800-61-RETEK (1-800-617-3835) World: + 1 612-630-5800
Fax	(+1) 612-630-5710
E-mail	support@retek.com
Internet	www.retek.com/support Retek's secure client Web site to update and view issues
Mail	Retek Customer Support Midwest Plaza 801 Nicollet Mall Suite 1100 Minneapolis, MN 55402

When contacting Customer Support:

- Always fill out an Issue Report Form before submitting issues to Retek (request forms from Customer Support if necessary).
- Provide a completely updated Customer Profile.
- Have a single resource per product responsible for coordination and screening of Retek issues.
- Respond to our requests for additional information in a timely manner.
- Use Retek Online Customer Support (ROCS) to submit and update your issues.
- Have a test system in place running base Retek code.

Contents

Chapter 1 – Introduction ... 1

Chapter 2 – Invoice Post [invcpost] ... 3

Modification ... 3

Design overview ... 3

Scheduling constraints ... 3

Restart recovery ... 4

Logical unit of work ... 4

Driving cursor ... 4

Program flow ... 5

Shared modules ... 5

Function level description ... 5

I/O specification ... 9

Technical issues ... 9

Chapter 3 – Promotion Price Extract and Download [prmext] ... 11

Modification ... 11

Design overview ... 11

Scheduling constraints ... 12

Restart recovery ... 12

Program flow ... 14

Shared modules ... 14

Function level description ... 14

I/O specification ... 14

Technical issues ... 14

Other ... 14

Chapter 1 – Introduction

This addendum to the Retek Merchandising System (RMS) 9.0.0.0 Operations Guide contains updates to the following batch designs:

- Invoice Post [invcpost]
- Promotion Price Extract and Download [prmext]

Refer to the following chapters for that information, which supercedes all comparable information in the RMS 9.0.0.0 Operations Guide. Each chapter contains a subsection indicating what specific modifications have been made.

Chapter 2 – Invoice Post [invcpost]

Modification

This program was modified to also post INVC_DISCOUNT records to the IIF_DISCOUNT table when posting an invoice.

Changed the mapping IIF_MATCH_DETAIL.SKU = INVC_MATCH_WKSHT.SHIPMENT to read IIF_MATCH_DETAIL.SKU = INVC_MATCH_WKSHT.SKU.

Design overview

This batch program will insert records into the IIF (invoice interface to financials) staging tables. It will insert new invoices with a payment status of 'H' (hold payment) so that the AP system can take all current/future liabilities into consideration. It will also insert approved matched invoices or force-paid invoices with a payment status of 'P' (ready to pay). Invoices from suppliers that are marked as pre-paid suppliers will be inserted with a payment status of 'U' (pre-paid unmatched) if unapproved or 'M' (pre-paid matched) if approved.

Approved invoices associated with approved debit/credit memos or credit notes will be inserted with a payment status of 'P' (ready to pay) since issuing a debit/credit memo should release the invoice for payment. However, the invoices and the debit/credit memos or credit notes associated should all be in approved status to be posted in tandem. Invoices with attached credit note requests will only be posted when an approved merchandise credit note has been attached. Credit note requests are never sent to financials. All types of invoices (except credit note requests), including debit/credit memos or credit notes, will be sent to the IIF staging tables even if they are not qualified for being posted.

Scheduling constraints

Processing Cycle:	daily
Scheduling Diagram:	after ediupinv and invmtch, before interface to financial system
Pre-Processing:	N/A
Post-Processing:	N/A
Threading Scheme:	None

Restart recovery

Logical unit of work

The logical unit of work for the invoice post module will be the invoice transaction. Restart/recovery will be based on the invoice transaction. In order to bundle an invoice and its associates, the invoice transaction here means the invc_id itself for an invoice not referencing another invoice and the ref_invc_id for an invoice that references another invoice. Therefore, one logical unit of work can contain multiple invoices.

Driving cursor

The driving cursor for this restart/recovery strategy looks like the following. Note that the second decode() of the ORDER BY clause is used to put an invoice ahead of its associates if it is referenced.

```
SELECT       ih.invc_id,
       ih.invc_type,
       ih.supplier,
       ih.partner_type,
       ih.partner_id,
       ih.ext_ref_no,
       ih.ref_invc_id,
       ih.ref_rtv_order_no,
       ih.ref_price_change,
       ih.ref_rsn_code,
       ih.terms,
       TO_CHAR(ih.due_date, 'YYYYMMDD'),
       ih.payment_method,
       ih.terms_dscnt_pct,
       ih.terms_dscnt_appl_ind,
       ih.freight_terms,
       TO_CHAR(ih.invc_date, 'YYYYMMDD'),
       ih.force_pay_ind,
       ih.post_date,
       ih.currency_code,
       ih.exchange_rate,
       ih.total_merch_cost,
       ih.total_qty,
       ih.comments,
       ih.status,
                ih.direct_ind,
                          ih.paid_ind,
                          ih.addr_key,
                          ih.payment_ref_no,
                          ih.payment_date,
                          ih.proof_of_delivery_no
          FROM  invc_head ih
         WHERE  ih.invc_type != 'R'
           AND  (ih.status = 'A'
            OR  (ih.force_pay_ind = 'Y'
                AND ih.status != 'P')
            OR  ih.post_date is NULL)
           AND  decode(ih.ref_invc_id,
                         NULL, ih.invc_id,
                       ih.ref_invc_id) >
       NVL(:ps_restart_invc_id, -999
       ORDER BY decode(ih.ref_invc_id, NULL, ih.invc_id,
       ih.ref_invc_id),
                      decode(ih.ref_invc_id, NULL, 0, 1);
```

The commit_max_ctr field on the restart_control table will determine the number of transactions that equal a logical unit of work. It should be set to prevent excessive rollback space usage. The recommended commit counter setting is 10000 records (subject to change based on experimentation).

Program flow

N/A

Shared modules

N/A

Function level description

init():

Retrieve system date and VAT indicator from PERIOD and SYSTEM_OPTIONS:

- vdate
- vat_ind

process():

In a LOOP, fetch the driving cursor. For each invoice fetched, insert into arrays (for insert into the IIF tables) as follows:

For the insert into IIF_HEAD, gather the following values for each invoice:

- invc_id = invc_head.invc_id
- invc_type = invc_head.invc_type
- supplier = invc_head.supplier
- partner_type = invc_head.partner_type
- partner_id = invc_head.partner_id
- ext_ref_no = invc_head.ext_ref_no
- payment_status will be decided by the action taken on this invoice, invc_head.direct_ind, and the prepay indicator:
  - Action is POST if
    - The invoice is being force-paid (force_pay_ind = 'Y')
    - The invoice is in 'A' status and not associated with any other invoices
    - The invoice is in 'A' status, associated with other invoices and all its associates are ready (debit memos/credit memos/credit notes are all in 'A' status, and, if any credit note request exists, debit memo or credit note in 'A' status exist).
  - The invoice is in 'A' status and referencing an invoice in 'P' status (either just posted or previously posted).
  - Action is SEND if the invoice is not qualified for being POSTed but has not been sent yet
  - Action is NONE if the invoice is not qualified for being POSTed and has been sent already
  - The prepay indicator indicates if a supplier is marked for pre-payment. It is fetched from sups.prepay_invc_ind for a supplier invoice; it is always 'N' for a non-supplier invoice
  - Payment_status is 'C' if invc_head.paid_ind is 'Y' for the invoice
  - Payment_status is 'M' if the action is POST and prepay indicator is 'Y'
  - Payment_status is 'P' if the action is POST and prepay indicator is 'N'
  - Payment_status is 'U' if the action is SEND and prepay indicator is 'Y'
  - Payment_status is 'H' if the action is SEND and prepay indicator is 'N'
- ref_invc_id = invc_head.ref_invc_id
- ref_rtv_order_no = invc_head.ref_rtv_order_no
- ref_price_change = invc_head.ref_price_change
- ref_rsn_code = invc_head.ref_rsn_code
- terms = invc_head.terms
- due_date = invc_head.due_date
- payment_method = invc_head.payment_method

- terms_dscnt_pct = invc_head.terms_dscnt_pct
- terms_dscnt_appl_ind = invc_head.terms_dscnt_appl_ind
- freight_terms = invc_head.freight_terms
- invc_date = invc_head.invc_date
- force_pay_ind = invc_head.force_pay_ind
- post_date = today's date
- currency_code = invc_head.currency_code
- exchange_rate = invc_head.exchange_rate
- total_payment_merch_cost = invc_head.total_merch_cost
- total_payment_qty = invc_head.total_qty
- comments = invc_head.comments
- addr_key = invc_head.addr_key
- payment_ref_no = invc_head.payment_ref_no
- payment_date = invc_head.payment_date
- proof_of_delivery_no = invc_head.proof_of_delivery_no
- direct_ind = invc_head.direct_ind

For the insert into IIF_NON_MERCH, gather the following values for each invoice:

- invc_id = invc_non_merch.invc_id
- non_merch_code = invc_non_merch.non_merch_code
- non_merch_amt = invc_non_merch.non_merch_amt
- vat_code = invc_non_merch.vat_code
- service_perf_ind = invc_non_merch.service_perf_ind
- store = invc_non_merch.store

For the insert into IIF_MERCH_VAT, gather the following values for each invoice:

- invc_id = invc_merch_vat.invc_id
- vat_code = invc_merch_vat.vat_code
- total_cost_excl_vat = invc_merch_vat.total_cost_excl_vat

For the insert into IIF_DETAIL, gather the following values for each invoice:

- invc_id = invc_detail.invc_id
- sku = invc_detail.sku
- payment_unit_cost = invc_detail.invc_unit_cost
- payment_qty = invc_detail.invc_qty

- payment_vat_rate = invc_detail.invc_vat_rate
- cost_dscrpncy_ind = invc_detail.cost_dscrpncy_ind
- qty_dscrpncy_ind = invc_detail.qty_dscrpncy_ind
- vat_dscrpncy_ind = invc_detail.vat_dscrpncy_ind

For the insert into IIF_MATCH_DETAIL, gather the following values for each invoice (invoice types 'C', 'D' and 'M' will not have records on this table):

- invc_id = invc_match_wksht.invc_id
- sku = invc_match_wksht.sku
- order_no = the order_no from SHIPMENT for invc_match_wksht.shipment
- asn_no = the ext_shipment from SHIPMENT for invc_match_wksht.shipment
- shipment = invc_match_wksht.shipment
- rcpt_date = the receive_date from SHIPMENT for invc_match_wksht.shipment

For invoices matched at the totals level, where there are no invc_match_wksht records, the values should be gathered from shipsku instead.

Once all IIF records are inserted into the array for the invoice, update the invoice and receipt header statuses appropriately.

- Set the post_date on INVC_HEAD to today's date.
- If the action taken for the invoice is POST, set the status on INVC_HEAD to 'P' (posted).
- If the invoice was in 'A' status, check each shipment matched to the invoice (retrieve using match_invc_id on SHIPSKU) to see if it is matched to any other invoices not in 'P' status. If it is, or if there are still SKUs on the shipment that are unmatched (match_invc_id is NULL), leave the shipment's invc_match_status as it is. If it is not, and all SKUs on the shipment are matched, set the shipment's invc_match_status to 'C'.
- End LOOP.

Only new columns shown

Table to Table

Source Table	Source Column	Field Type
INVC_DISCOUNT	invc_id	number(10)
	seq_no	number(6)
	discount_type	varchar2(6)
	discount_value	number(20,4)
	applies_to_amt	number(20,4)

Target Table	Target Column	Calculations
IIF_DISCOUNT	invc_id	None
	seq_no	None
	discount_type	None
	discount_value	None
	applies_to_amt	None

process_invc()

After the calls to insert_iif_non_merch() and insert_iif_merch_vat(), and before the call to insert_iif_detail(), make a call to the new function insert_iif_discount()

insert_iif_discount()

Perform a select insert to move all data from INVC_DISCOUNT to IIF_DISCOUNT for the invoice in the pa_invc_array current record

I/O specification

N/A

Technical issues

N/A

Chapter 3 – Promotion Price Extract and Download [prmext]

Modification

Added a list of valid tran_types for the price_hist table.

Design overview

The prmext program extracts promotions from the promotion master tables within the Retek system and sends promotion price details to the point of sale system. Additionally, promotional price history is stored for each valid SKU/store combination. When a store is within the POS threshold extraction date, all SKUs on the promotion will be extracted to the store (provided the item is stocked at the store). Additionally, the prmext program has the ability to extract promotion SKU changes throughout the life of a promotion based on the promsku status. The promsku status will provide an indication of SKU details that have not been extracted. (Possible changes are new SKU / deleted SKU / changed promotion price.) The SKU will be re-extracted to stores that are currently active with the given promotion. Stores that are to be extracted will not differentiate between promotion SKU changes since all SKUs will be extracted to the POS, provided the SKU has not been deleted from the promotion.

The promotional retail is stored in the history tables in the local currency of the store and this is also the price that is transmitted to POS. Since the extraction process is performed at a promotion store level, this allows different stores to be effectively on promotion for varying time frames. If the promotion start is within pos_extract_days from tomorrow, the promotion store will be extracted to the point of sale, and a price history record is written with a future action date. The status of the promotion store is updated to extracted. Once all stores on a promotion have been extracted the overall status of the promotion on the header is set to extracted. If the promotion store end date is within pos_extract_days from tomorrow, the regular price will be extracted to the POS and the promotion store status will be updated to completed. Once all store have completed, the overall promotion header status is updated to completed.

For each promotion which is due to start or end within the number of days in pos_extract_days from UNIT_OPTIONS, a POS_MODS row is built containing the details necessary for the POS PLU update for each item (SKU) included in the promotion. Note that if a fashion style has been included in a promotion, it must be expanded to its component SKUs (sizes and colors).

This program checks overlap with price changes if an overlap is found, it does not insert price_hist, but still inserts pos_mods, with the price fetched from the price change tables.

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
UNIT_OPTIONS	No	Yes	No	No	No
PERIOD	No	Yes	No	No	No

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
PROMHEAD	Yes	Yes	No	Yes	No
PROMSTORE	Yes	Yes	No	Yes	No
PROMSKU	Yes	No	No	No	No
WIN_SKUS	Yes	Yes	No	No	No
RAG_STYLE	Yes	Yes	No	No	No
RAG_SKUS	Yes	Yes	No	No	No
PACKHEAD	Yes	Yes	No	No	No
POS_MODS	No	No	Yes	No	No
PRICE_HIST	No	No	Yes	No	No
V_RESTART_STORE	No	Yes	No	No	No
PRICE_SUSP_HEAD	No	Yes	No	No	No
PRICE_SUSP_DETAIL	No	Yes	No	No	No

Scheduling constraints

Processing Cycle: PHASE 1 – (daily)

Scheduling Diagram: N/A

Pre-Processing: N/A

Post-Processing: Post processing for prmext resets all promotion SKU status after the SKU has been extracted (or re-extracted) to a store. Since stores have various start and stop dates and promotion maintenance can occur at any point during a promotion, the prepost program resets the change status on promsku to ensure that the same change is not forwarded down to POS more than once.

Threading Scheme: STORE

V_restart_store

Restart recovery

```
SELECT ph.promotion,
       ph.event,
           TO_CHAR(ps.start_date, 'DDMMYYYY'),
           TO_CHAR(ps.end_date, 'DDMMYYYY'),
           ps.store,
           0,                /* promotion action type */
           decode(ps.extract_status,’E’,1,0),
           ph.rowid,
           ps.rowid
      FROM v_restart_store rv,
           promstore ps,
           promhead ph
     WHERE ph.promotion = ps.promotion
       AND ps.start_date<=
to_date(:vdate,'DDMMYYYY')+:pos_extract_days
       AND ph.status in ('A','E')
       AND nvl(ps.extract_status, ‘E’) = ‘E’
       AND rv.driver_value = ps.store
       AND rv.driver_name = :ora_restart_driver_name
       AND rv.num_threads = :ora_restart_num_threads
       AND rv.thread_val  = :ora_restart_thread_val
       AND (ps.promotion > NVL(:ora_restart_promotion, -
999) OR
               (ps.promotion = :ora_restart_promotion
AND
                   (ps.store >= :ora_restart_store)))
 UNION ALL
    SELECT ps.promotion,
           ph.event,
           TO_CHAR(ps.start_date, 'DDMMYYYY'),
           TO_CHAR(ps.end_date, 'DDMMYYYY'),
           ps.store,
           1,                /* promotion action type */
           0,                /* set to not extracted */
           ph.rowid,
           ps.rowid
      FROM v_restart_store rv,
           promstore ps,
           promhead ph
     WHERE ph.promotion = ps.promotion
       AND ps.end_date <= to_date(:vdate,'DDMMYYYY')+
:pos_extract_days
       AND (ps.extract_status = 'E'
        OR (ps.extract_status is NULL AND ps.end_date =
ps.start_date))
       AND rv.driver_value = ps.store
       AND rv.driver_name = :ora_restart_driver_name
       AND rv.num_threads = :ora_restart_num_threads
```

```
           AND rv.thread_val  = :ora_restart_thread_val
           AND (ps.promotion > NVL(:ora_restart_promotion, -
    999) OR
                   (ps.promotion = :ora_restart_promotion
    AND
                     (ps.store >= :ora_restart_store)))
      ORDER BY 1,5,6;
```

Program flow

N/A

Shared modules

GET_SYSTEM_IND: fetches the merchandise type for the SKU to be processed from the desc_look table.

Function level description

N/A

I/O specification

N/A

Technical issues

N/A

Other

The price_hist.tran_type contains a code number that indicates the type of transaction that caused the price change. Valid values are:

0 = New item added

2 = Unit cost was changed

4 = Single unit retail was changed

8 = Single unit retail was changed in Clearance

9 = Single unit retail was changed in Promotion

10 = Multi-unit retail was changed

11 = Single-unit retail and Multi-unit retail were changed

99 = Item was deleted from file