Retek® Merchandising System 9.0.11

Addendum to Operations Guide

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

Introduction

This addendum to the Retek Merchandising System (RMS) 9.0 Operations Guide contains updates to the following batch designs:

- Transfer Shipments Upload [tsfoupld]

Transfer Shipments Upload [tsfoupld]

Design overview

The purpose of this batch module is to accept transfer shipment details from an external system. The transfer transactions will provide feedback to existing transfers within the Retek system or initiate manual transfers created in an external system. The following functions will be performed for each transferred item :

- create/update transfer and shipment header and detail records.
- create item/location relation for receiving location (if it doesn't exist)
- update perpetual inventory and in transit qtys for source location
- update the average cost of item and in transit qtys for receiving location
- write financial transactions for both the transfer out and the transfer in
- update stock count's snapshot on hand quantity for source location and snapshot in transit quantity for destination location if stock count is in progress
- create/update bill of lading
- create/update warehouse issues history (if transfer from a warehouse to a store)
- update unavailable inventory status quantity for NS (Non-salable) type of transfer for source location
- update quantity transferred on allocation detail table if this transfer was created from a standalone allocation

TABLE	INDEX	SELECT	INSERT	UPDATE	DELETE
TSFHEAD	No	Yes	Yes	Yes	No
TSFDETAIL	No	Yes	Yes	Yes	No
SHIPMENT	No	Yes	Yes	Yes	No
SHIPSKU	No	Yes	Yes	Yes	No
POS_MODS	No	No	Yes	No	No
PRICE_HIST	No	No	Yes	No	No
RAG_SKUS_ST	No	Yes	No	Yes	No
WIN_STORE	No	Yes	No	Yes	No
RAG_SKUS_ST	No	Yes	No	Yes	No
WIN_WH	No	Yes	No	Yes	No
TRAN_DATA	No	No	Yes	No	No
RAG_SKUS	No	Yes	No	No	No
RAG_STYLE_ST	No	Yes	No	No	No
RAG_STYLE_WH	No	Yes	No	No	No
INV_STATUS_QTY	No	Yes	No	Yes	Yes
INV_STATUS_TYPES	No	Yes	No	No	No

Scheduling constraints

Processing Cycle: PHASE 2 (daily)

Scheduling Diagram: This program must run before the transfer in batch module and will likely be run at the beginning of the batch run during the POS polling cycle, or possibly at the end of the batch run if pending warehouse transactions. It can be scheduled to run multiple times throughout the day, as WMS or POS data becomes available. In a true DC flow through type of operation, this program should also be run after Carton Receiving Upload (ctniupld) module to ship the cross-dock carton transfers created in ctniupld so that the goods received into DC for a cross-dock PO are shipped out to the final destination within the same day.

Pre-Processing: N/A

Post-Processing: N/A

Threading Scheme: STORE and WH

Threads driven by number of distinct files

Restart recovery

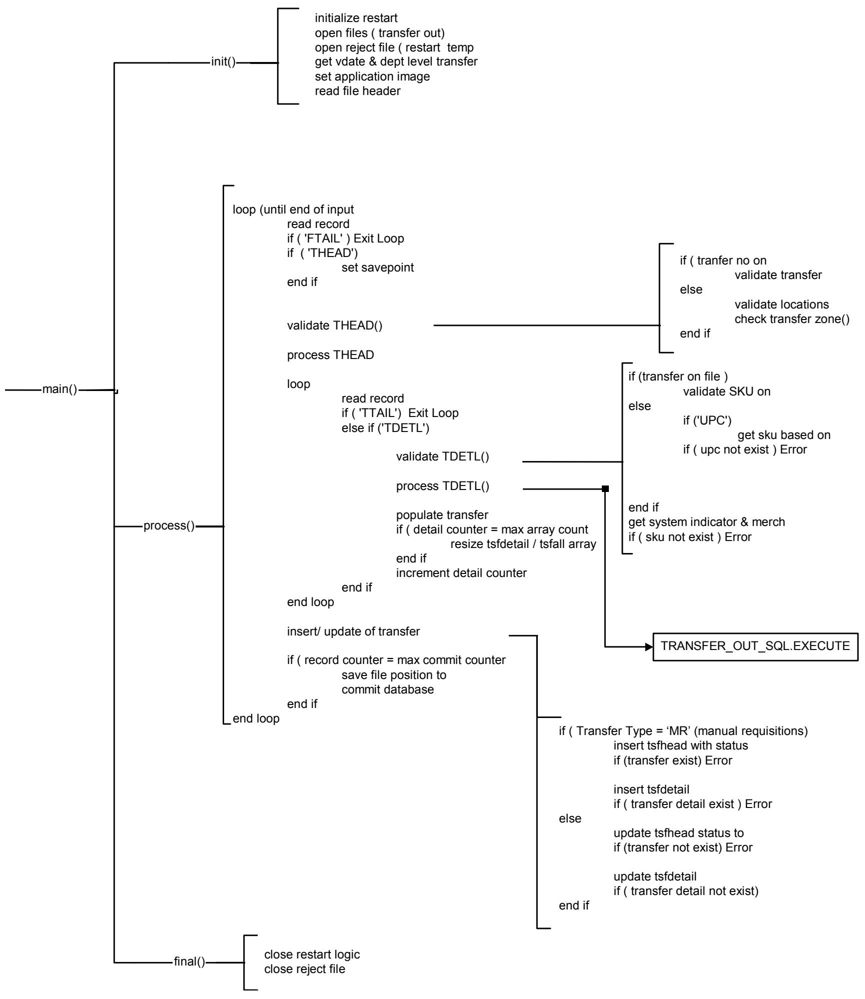
The logical unit of work for the transfer out module is the discrete transfer transaction. Each transfer will be identified by the transfer number (if it already exists in the Retek system) or a unique transaction set number generated by the external system. This transfer transaction will be defined as the logical unit of work. If any portion of the processing for the complete transfer transaction fails, the entire transfer must be re-processed.

A savepoint will be issued prior to processing a new transfer. If any record within the transaction fails, the whole transaction will be rolled back to the most recent savepoint. This way, the successfully processed transactions will remain posted to the database but not yet committed.

To prevent excessive rollback space usage, intermittent commits will be issued based on a commit counter. The recommended commit counter setting is 10000 records (subject to change based on experimentation). The commit counter is based on actual records processed, not overall transactions, nor the number of writes to the database, since the database interactions will be a constant multiplier of the commit counter. A transfer transaction cannot be committed to the database until it is complete so the commit counter is viewed as a minimum threshold that once reached, will force a commit after the completion of the current transfer transaction.

Error handling will be based on the logical unit of work also. If a given record within a transfer transaction fails, that error will be posted to the standard error log for the batch module. If the error is of a non-fatal type, all subsequent detail records within the transfer will continue to be processed and any errors noted will continue to be posted. After processing all errors for the transaction, the entire transfer will be rejected to a runtime specified rejection file. If a fatal error is encountered, the file pointer at the time of the last commit will have been posted to the bookmark and all transactions from the last commit will be rolled back. Processing will commence with from the saved file position.

Program flow



```
main()
  init()
      initialize restart
      open files ( transfer out)
      open reject file ( restart  temp
      get vdate & dept level transfer
      set application image
      read file header

  process()
      loop (until end of input
              read record
              if ( 'FTAIL' ) Exit Loop
              if  ( 'THEAD')
                      set savepoint
              end if

              validate THEAD()
                      if ( tranfer no on
                              validate transfer
                      else
                              validate locations
                              check transfer zone()
                      end if

              process THEAD

              loop
                      read record
                      if ( 'TTAIL')  Exit Loop
                      else if ('TDETL')

                              validate TDETL()
                                      if (transfer on file )
                                              validate SKU on
                                      else
                                              if ('UPC')
                                                      get sku based on
                                              if ( upc not exist ) Error

                                      end if
                                      get system indicator & merch
                                      if ( sku not exist ) Error

                              process TDETL()  ->  TRANSFER_OUT_SQL.EXECUTE

                              populate transfer
                              if ( detail counter = max array count
                                      resize tsfdetail / tsfall array
                              end if
                              increment detail counter
                      end if
              end loop

              insert/ update of transfer
                      if ( Transfer Type = 'MR' (manual requisitions)
                              insert tsfhead with status
                              if (transfer exist) Error

                              insert tsfdetail
                              if ( transfer detail exist ) Error
                      else
                              update tsfhead status to
                              if (transfer not exist) Error

                              update tsfdetail
                              if ( transfer detail not exist)
                      end if

              if ( record counter = max commit counter
                      save file position to
                      commit database
              end if
      end loop

  final()
      close restart logic
      close reject file
```

Shared modules

TRANSFER_OUT_SQL.EXECUTE: Package referenced to perform transfer out logic, including

- create item/location relation for receiving location (if it doesn't exist)
- update perpetual inventory for source location
- update the average cost of item for receiving location
- write financial transactions for both the transfer out and the transfer in
- update stock count's snapshot on hand quantity for source location and snapshot in transit quantity for destination location if stock count is in progress
- create/update bill of lading
- create/update warehouse issues history (if transfer from a warehouse to a store)

- update unavailable inventory status quantity for NS (Non-salable) type of transfer for source location
- update quantity transferred on allocation detail table if this transfer was created from standalone allocation

TRANSFER_IN_SQL.EXECUTE: Package referenced to perform transfer in logic for customer order types of transfers where the delivery type for the transfer is 'Ship Direct.'

- update perpetual inventory for destination location
- update stock count's snapshot on hand quantity for destination location if stock count is in progress
- update unavailable inventory status quantity for NS (Non-salable) type of transfer for destination location
- update perpetual inventory with adjustments for detailed receipt discrepancies and create stock ledger stock adjustment transactions, if system_options.auto_close_tsf = 'Y'

The following are called from TRANSFER_OUT_SQL and/or TRANSFER_IN_SQL packages and are thus, indirect calls.

STOCK_LEDGER_SQL.TRAN_DATA_INSERT: Package referenced by TRANSFER_OUT_SQL.EXECUTE to perform the stock ledger transaction inserts for the transfer out of the goods from the source location and the transfer in of the goods at the destination location.

NEW_STAPLE_LOC, NEW_FASHION_LOC, NEW_PACK_LOC: These stored procedures are used to create item/location relationships for locations that are to receive goods on a transfer and have not yet stocked the given item.

INVADJ_SQL.ADJ_UNAVAILABLE0: called to update the unavailable inventory status quantity

INVADJ_SQL.ADJ_TRAN_DATA : called to write tran_data record for unavailable inventory adjustment

Function level description

init()

```
declare structure arrays for tsfdetail
initialize restart recovery

open input file ( transfer out )
        - file should be specified as input parameter to program
open reject file ( as a temporary file for restart )
        - file should be specified as input parameter to program

get vdate and department level transfer indicator from period table and system options
set application image array - save the line counter
read file header record
```

if (record type <> ‘FHEAD’)   Fatal Error

---

process()

```
loop
        read record from input file
        if ( 'FTAIL' )
                Exit Loop
        end if
        if   ( 'THEAD')
                set savepoint and save current file pointer position
                validate_THEAD()

                reset detail count
                process_THEAD()
        end if

        loop
                check carton flag to determine if tdetl records will be for a carton or not
                read record from input file (different structure for carton or regular)
                if ( 'TTAIL')   Exit Loop
                if ('TDETL')
                        validate_TDETL()
                        process_TDETL()
                end if

                if ( detail count = max array count )
                        resize array structures for tsfdetail
                        increase max array count
                end if
                increment detail count
        end loop

        if ( no errors   )
                post_transfers() (don’t call this if doing a carton)
        end if
        if ( non Fatal Error Encountered )
                reject_record   - call write error and pass file pointer as of last savepoint
                and current file pointer
                Rollback transaction
        end if

        if ( transaction count > max commit count )
                restart file commit
                        - save the current input file pointer position
```

```
                    - save the line counter in restart image
        end if
end loop

restart commit final
```

<u>validate_THEAD()</u>

```
- validate transfer
-if external shipment number is 'CARTON', set carton flag and return   from function


format_header_fields()

if   ( shipment number provided in transaction )

        validate that the shipment number exists within Retek for a transfer. (check on
        shipment)
        validate that the transfer within Retek has a status of 'A', 'E', 'S', 'C' (approved,
extracted, shipped, closed) and is applicable to the
                to/from locations specified   (check on tsfhead) – also fetch transfer type

        if shipment number provided does not exist on shipment in 'I', 'R' status for a
        transfer then
                raise Non-Fatal Error
        if transfer does not exist in Retek with the appropriate status and locations then
                raise Non-fatal error

else if ( no shipment number is provided )
        if (external shipment number provided)
                - validate to and from locations
                if ( loc_type = 'S' )
                        check for existence on store table
                else ( loc_type = 'W' )
                        check for existence on wh table
                end if
                if any location not exist, write non-Fatal error

                - validate common transfer zone for store to store transfers
                if ( to_loc type = 'S' and from_loc = 'S')
                        check transfer zone - select   transfer zone of the from location and
                the to
                        location.
                        if ( from_loc transfer zone <> to _loc transfer zone )
                                write non-Fatal Error ( transfer zones incompatible )
                end if
            end if
        else (no external shipment number)
            All detail records must have a allocation number.
        end if
end if
```

<u>process_THEAD()</u>

```
check for a bill of lading in 0 - open status for the destination location
retrieve the bill of lading number if one exists
if ( bill of lading does not exist )
    get next bill of lading number
    insert bill of lading header ( lad_head ) record
end if

if bol number passed in   ensure it is valid.
If it is not valid get next bol number.

if transfer type = 'CO'
    retrieve delivery type from the ORDCUST table
end if
```

<u>validate_TDETL()</u>

```
format_detail_fields()

if inventory status field is not blank, validate it against inv_status_types table

if no shipment / ext shipment in file
    every detail line must have an allocation.

if (shipment number in file )
    validate item exists on the transfer
else
    if ( Item Type = 'UPC' )
        select sku from upc_ean based on the upc and supplement
        if ( upc does not exist )
            write non-Fatal Error ( upc not found )
        end if
    else if ( Item Type = 'SKU' )
        SKU = item value from the input file
        case ID = ' '
    end if
end if
```

if the store rcv type is ‘C’ the carton field must be populated

```
- get item system indicator, department, class and subclass
if ( system indicator does not exist )
        write non-Fatal Error ( sku not found )
end if
```

---

process_TDETL()

The upd_resv_ind and the upd_intran_ind should be setup in the following way before calling transfer_out_sql.execute.

```
      if :oi_new_tsf_flag = 1 then
         if :os_store_rcv_type = 'A' then
            L_upd_resv_ind    := 'N';
            L_upd_intran_ind := 'N';
         else
            L_upd_resv_ind    := 'N';
            L_upd_intran_ind := 'Y';
         end if;
     elsif :ora_tsf_type = 'CO' and :ora_deliver_type = 'S' or
            :os_store_rcv_type = 'A' then
         L_upd_resv_ind    := 'Y';
         L_upd_intran_ind := 'N';
      else
         if :os_tsf_status = 'C' then
            L_upd_resv_ind    := 'N';
         else
            L_upd_resv_ind    := 'Y';
         end if;
         L_upd_intran_ind := 'Y';
      end if;
```

```
call TRANSFER_OUT_SQL.EXECUTE package function
(see design specification for TRANSFER_OUT_SQL)

if transfer type = ‘CO’ and delivery type = ‘S’ or store receive type is ‘A’
    call TRANSFER_IN_SQL.EXECUTE package function
    (see design specification for TRANSFER_IN_SQL)

write_recs_to_struct()
```

---

post_transfers()

```
if ( shipment number was not passed in on the input file )
        insert TSFHEAD   (transfer_type = ‘MR’ or PO in an allocation is passed in,
        ext_ref_no = external shipment number)
        insert SHIPMENT   (ext_ref_no_out should be the transaction control number,
        ship date should be the transaction date)
        perform array insert of TSFDETAIL
        perform array insert of SHIPSKU
else (  for all other Retek initiated transfer transactions )
        try to update shipsku record if no data is found
        perform array update of TSFDETAIL, set ship_qty – if transfer type = ‘SA’, set
        tsf_qty = 0
        perform array insert of SHIPSKU

        - The this transfer is a customer order (tsf_type = ‘CO’) with a delivery type of
        direct ship to customer, then this transfer must also be closed when it is sent.

        if transfer type = ‘CO’ and delivery type = ‘S’ or store rcv type is ‘A’
                call TRANSFER_IN_SQL.CLOSE
                (see design specification for TRANSFER_IN_SQL)
        else if transfer type = ‘SA’ then
                update TSFHEAD status to ‘A’ - approved
        else
                update TSFHEAD status to ‘S’ - shipped
        end if
end if
```

<u>format_header_fields()</u>

```
assign input file fields to variables

if from location type = ‘ST’
        set ora_from_type = ‘S’
else if from location type = ‘WH’
        set ora_from_type = ‘W’
end if

if to location type = ‘ST’
        set ora_to_type = ‘S’
else if to location type = ‘WH’
        set ora_to_type = ‘W’
end if
```

<u>format_detail_fields()</u>

assign input file fields to variables

- transfer quantity has an implied 4 decimal places
transfer qty = transfer qty / 10000

---

process_carton()
Select details from transfer tables for the carton number; for each sku in the carton, call process_TDETL.

ON Fatal Error

- rollback to last physical commit point
- Exit Program

ON Non-Fatal Error

- rollback to last savepoint
- write out complete transfer transaction to the reject file, pass file pointer at last savepoint and current file pointer

I/O specification

Input File

The input file should be accepted as a runtime parameter at the command line.

IMPORTANT:

The structure of the TDETL line will vary, depending on whether cartons are included or not. If cartons are included, the line will end after the item value field.

Record Name	Field Name	Field Type	Default Value	Description
File Header	File Type Record Descriptor	Char(5)	FHEAD	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Line number of the current file
	File Type Definition	Char(4)	TSFO	Identifies file as 'Transfer OUT'
	File Create Date	Date	create date	date file was written by external system
Transaction Header	File Type Record Descriptor	Char(5)	THEAD	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Line number of the current file
	Transaction Set Control Number	Number(14)	specified by external system	used to force unique transaction check
	Transaction Date	Date	specified by external system	date the transfer was created in external system
	From Location Type	Char(2)	ST - store WH - warehouse	specifies the type of location sending items
	From Location Value	Number(4)	location identifier	Specifies the sending location id number
	To Location Type	Char(2)	ST - store WH - warehouse	specifies the type of location receiving items
	To Location Value	Number(4)	location identifier	Specifies the receiving location id number

Record Name	Field Name	Field Type	Default Value	Description
	Shipment Number	Number(10)	Retek shipment number	specifies the Retek shipment cross-reference
	External shipment	Char(15)	External shipment number	specifies external shipment number; will be CARTON when transferring cartons
	Courier	Char (20)	Courier used to ship order	
	Arrival date	Date	Arrival date	
	Number of boxes	Number(4)		Number of boxes in this transfer
	BOL number	Number(13)	Bill of lading	
Transaction Detail (Item)	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Line number of the current file
	Transaction Set Control Number	Number(14)	specified by external system	used to force unique transaction check
	Detail Sequence Number	Number(6)	specified by external system	sequential number assigned to detail records within a transaction
	Item Type	Char(3)	UPC SKU	item type will be represented as a UPC or SKU
	Item Value	Number(13)	item identifier	the id number of a SKU or UPC
	Supplement	Number(5)	supplemental identifier	used to further specify the id of an UPC item
	Allocation Number	Char(6) or char(10) if the allocation_ind is = ‘Y’.	allocation identifier	Retek allocation number attached to the transfer
	Inventory Status	Number(2)	inventory status of item	used to indicate the type of non-salable merchandise transferred in an ‘NS’ transfer

Record Name	Field Name	Field Type	Default Value	Description
	carton	Char(20)	carton identifier	UCC – 122 carton code
	Transfer Quantity	Number(12)		number of units to be transferred of the given item (*10000—4 implied decimal places)
Transaction Detail (Carton)	File Type Record Descriptor	Char(5)	TDETL	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Line number of the current file
	Transaction Set Control Number	Number(14)	specified by external system	used to force unique transaction check
	Detail Sequence Number	Number(6)	specified by external system	sequential number assigned to detail records within a transaction
	Item Type	Char(3)	CTN	item type will be represented as a CTN when transferring a carton
	Item Value	Char(20)	carton identifier	UCC – 122 carton code
Transaction Trailer	File Type Record Descriptor	Char(5)	TTAIL	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Line number of the current file
	Transaction Detail Line Count	Number(6)	sum of detail lines	sum of the detail lines within a transaction
File Trailer	File Type Record Descriptor	Char(5)	FTAIL	Identifies file record type
	File Line Sequence	Number(10)	specified by external system	Current line number
	Number of transaction lines	Number(10)	specified by external system	total number of lines in file, excluding FHEAD and FTAIL

Output File

Record Name	Field Name	Field Type	Default Value	Description
	Record Type	Char (1)	H	Specifies file record type
	Store Order Number	Number (10)	Order No	Specifies shipment number
	Division Type	Char (2)	Division Type	Specifies division type
	Warehouse	Number (6)	WH Loc	Specifies WH location value
	Store	Number (6)	Store Loc	Specifies ST location value
	Store Order Type	Number (4)	Store order type	Specifies transfer type
	Store order comment	Char (255)	Comment	Specifies store order comment (from shipment or transfer or both)
	Ship Date	Number (14)	Ship date	Specifies date shipped (date when file was processed + 1)

Detail

Record Name	Field Name	Field Type	Default Value	Description
	Record Type	Char (1)	D	Specifies record type
	Store Order number	Number (10)	Order No	Specifies Shipment Number
	Division type	Char (2)	SA, PO, MR, CO, AD	Specifies Division Type
	Xref Div Item	Number (8)		RMS SKU
	UPC	Number (13)	UPC value	Specifies UPC Value
	UPC supplement	Number (5)	UPC supplement	Specifies UPC supplement value
	Unit of Measure	Char (4)	Unit of Measure	Specifies unit of measure

Record Name	Field Name	Field Type	Default Value	Description
	SKU Deck Cost	Number (10)	Deck cost	Average unit cost
	Quantity Shipped	Number (12)	Quantity Shipped	Specifies quantity shipped value

Reject File

The reject file should be able to be re-processed directly. The file format will therefore be identical to the input file layout. The file header and trailer records will need to be created by the transfer out module and a reject line counter will be required to ensure that the file line count in the trailer record matches the number of rejected records. A reject file will be created in all cases. If no errors occur, the reject file will consist only of a file header and trailer record and the file line count will be equal to 0.

The reject filename should also be specified as a runtime parameter.

Error File

Standard Retek batch error handling modules will be used and all errors (fatal & non-fatal) will be written to an error log for the program execution instance. These errors can be viewed on-line with the batch error handling report.

Technical issues

N/A