# swiftprep Group

John D. Baker (john.baker@jacksons.com)

https://bitbucket.org/jacksonsbidevs/swiftiq_staging/src/master/SwiftCode/swiftprep.ijs

SHA-256: 3763408d3b6586feb184a3178e78fb492bf4b3e257f328ffad267025674e3391

August 5, 2021

## Contents

# swiftprep Overview

swiftprep is a J script that converts zipped SwiftIQ *.csv files to standard TAB delimited *.txt files that are then loaded to the SqlServer database SwiftIQ_Staging on server JFSDEV04 with the SSIS package swiftiq_staging.ispac. **The SSIS package is invoked from J.** Having J control SSIS results in a more agile system than the other way around.

swiftprep is typically invoked from a scheduled batch script:

```
\\jfsdev04\shares\SwiftIQPreprocess\SwiftCode\swiftLoadLast.bat
```

swiftprep makes a number of assumptions:

1. Users have file permissions to: \\jfsdev04\shares\SwiftIQPreprocess
2. ODBC dsns swift_staging and swift_reporting have been defined on the local machine.
3. Users have access to the SQLServer database SwiftIQ_Staging on JFSDEV04.
4. Users have access to the SQLServer database SwiftIQ_Reporting on 192.168.2.102.

All swiftprep related code is in the repository:

https://bitbucket.org/jacksonsbidevs/swiftiq_staging/src/master/

## swiftprep Interface

```
dailyswift          [20] daily SwiftIQ zip loading and maintenance tasks
fullheaderscan      [23] scans all the SwiftIQ zips and builds complete table headers
loadswiftcsvs       [26] copies zip to SwiftIQ working directory and extracts csvs
loadswifttabs       [27] convert SwiftIQ csvs to TAB delimited table files
```

## `swiftprep` ETL Notes

`swiftprep` transforms source `*.csv` data in the following ways:

1. All quoted comma characters, (`"quoted , ehh"`) are convertd to semicolons (`"quoted ; ehh"`).
2. Tab and double quote `"` characters are removed before parsing lines.
3. Surrounding white space, i.e. leading and trailing blanks, carriage returns, line feeds, and tabs are removed. If any leading blanks are significant `"  blanks matter here"` this will damage data. In the vast majority of cases surrounding white space is *depraved garbage* so `swiftprep` takes out the trash.

## swiftprep Source Code

```
NB.*swiftprep s-- load SwiftIQ zipped CSV data.
NB.
NB. verbatim: interface word(s):
NB. ------------------------------------------------------------------------
NB.  dailyswift          - daily SwiftIQ zip loading and maintenance tasks
NB.  fullheaderscan      - scans all the SwiftIQ zips and builds complete table headers
NB.  loadswiftcsvs       - copies zip to SwiftIQ working directory and extracts csvs
NB.  loadswifttabs       - convert SwiftIQ csvs to TAB delimited table files
NB.  procqueuezips       - processes queued zips from file
NB.  procswiftzip        - process a SwiftIQ zip
NB.  queueswiftSizes     - file sizes of SwiftIQ queues: queueswiftSizes SwiftAuxDir,'swift2.txt'
NB.  queueswiftZips      - write (y) line file of queued zips
NB.  runswiftssis        - runs SwiftIQ SSIS batch scripts
NB.  showswiftUnload     - list never loaded zips in load order
NB.  showswiftlog        - show SwiftIQ log entries
NB.  swiftLoaded         - loaded swift zips with return code: swiftLoaded '<1000'
NB.  swiftMissingColumns - list missing SwiftIQ landing table columns
NB.  swiftZips           - sorted SwiftIQ zip files
NB.
NB. author:  John D. Baker -- john.baker@jacksons.com
NB. created: 2021may13
NB. ------------------------------------------------------------------------
NB. 21jul02 (runswiftssis) added
NB. 21jul07 (fullheaderscan) added
NB. 21jul08 (swiftMissingColumns) added
```

```
NB. 21jul09 (procqueuezips, showswiftlog) added
NB. 21jul14 (queueswiftZips) added
NB. 21jul15 (queueswiftSizes) added
NB. 21jul16 (appendswiftcsv) added
NB. 21jul20 (showswiftUnload, swiftLoaded) added
NB. 21jul27 (dailyswift) added
NB. 21jul28 (cntswiftrows) added

require 'task dd'
coclass 'swiftprep'

NB.*dependents
NB. (*)=: LogSwiftIQ_sql RepCntSwiftIQ_sql StgCntSwiftIQ_sql
NB.*enddependents

LogSwiftIQ_sql=:  (0 : 0)
with
  runsecs
  as
  ( select s.TaskID, datediff(ss, s.LoadUtcDateTime, e.LoadUtcDateTime) as LoadSeconds
    from log_EtlTasks s
      left join log_EtlTasks e on s.TaskID = e.TaskID
        and e.TaskItem = '::END>> swift csv zip load'
    where datediff(ss, s.LoadUtcDateTime, e.LoadUtcDateTime) <> 0)
select {~{~TOPN~}~}
  a.LoadUtcDateTime
  , a.TaskID
```

```
  , b.TaskRC
  , isnull(runsecs.LoadSeconds, 0) as LoadSeconds
  , b.TaskItem
  , a.LoadZipName
  , a.LoadMessage
from log_LoadedFiles a
  inner join log_EtlTasks b on a.TaskID = b.TaskID
  left join runsecs on a.TaskID = runsecs.TaskID
where b.TaskItem like '::END>>%' and len(a.TaskID) > 0 and b.TaskRC {~{~RC~}~}
order by a.LoadUtcDateTime desc
)

RepCntSwiftIQ_sql=:  (0 : 0)
select count(stage_ItemSalesKey) as RepCnt from ItemSales
union all
select max(stage_ItemSalesKey) as RepMax from ItemSales
)

StgCntSwiftIQ_sql=:  (0 : 0)
select count(stage_ItemSalesKey) as StgCnt from stage_ItemSales
union all
select count(stage_ItemSalesKey) as StgDif from stage_ItemSales
where {~{~RKEY~}~} < stage_ItemSalesKey
)

NB.*end-header

NB. carriage return character
```

```
CR=: 13{a.

NB. carriage return line feed character pair
CRLF=: 13 10{a.

NB. when 1 check prior load when 0 ignore and reload
CheckPriorSwiftZip=: 1

DTExecOKMsg=: 'DTExec: The package execution returned DTSER_SUCCESS (0).'

NB. file header sample bytes
HEADERLEN=: 10000

NB. interface words (IFACEWORDSswiftprep) group
IFACEWORDSswiftprep=: <;._1 ' dailyswift fullheaderscan loadswiftcsvs loadswifttabs procqueuezips procswift
>..>zip queueswiftSizes queueswiftZips runswiftssis showswiftUnload showswiftlog swiftLoaded swiftMissingColum
>..>ns swiftZips'


NB. location of J unzip.exe
JunzipCmd=: 'c:\j64\j902\tools\zip\unzip.exe'

NB. line feed character
LF=: 10{a.

NB. log message prefix
LogMsgPfx=: '>-<'
```

```
NB. maximum number of SQL column LoadMessage characters
LogTextLim=: 1999

NB. ODBC dsn connection string for swift staging
ODBCSWIFT=: 'dsn=swift_staging'

NB. ODBC dsn connection string for swift reporting
ODBCSWIFTREPORT=: 'dsn=swift_reporting'

NB. root words (ROOTWORDSswiftprep) group
ROOTWORDSswiftprep=: <;._1 ' IFACEWORDSswiftprep LogSwiftIQ_sql ROOTWORDSswiftprep RepCntSwiftIQ_sql StgCnt
>..>SwiftIQ_sql dailyswift fullheaderscan logswiftLast procqueuezips queueswiftSizes queueswiftZips shard show
>..>swiftUnload swiftMissingColumns'


NB. flag that controls SSIS package execution default value = 1
RUNSSISFLAG=: 1

NB. valid SwiftIQ staging column name characters
SWIFTALPHA=: ',_ 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz'

SwiftAuxDir=: '\\jfsdev04\Shares\SwiftIQPreprocess\SwiftAux\'

SwiftCodeDir=: '\\jfsdev04\Shares\SwiftIQPreprocess\SwiftCode\'

SwiftErrorLogFile=: 'aa_ERROR_SwiftIQ.txt'
```

```
NB. indexed file read byte limit
SwiftIreadBytes=: 25000000

NB. complete standarized SwiftIQ landing table column names
SwiftLandTableHeaders=: 5 2$(<'Calendar'),(<;._1 ' cal_key cal_fiscal_year cal_fiscal_year_begin cal_type
>..>cal_custom_periods ts'),(<'FuelProduct'),(<;._1 ' product_key upc_sell_unit_desc upc sell_unit_desc sell_
>..>unit_qty item_id item_desc sub_category_id sub_category_desc category_id category_desc department_id depar
>..>tment_desc size_desc item_type grp_level_4_id grp_level_4_desc alt_grp_level_1_id alt_grp_level_1_desc alt
>..>_grp_level_2_id alt_grp_level_2_desc alt_grp_level_3_id alt_grp_level_3_desc create_date ent_item_number u
>..>pc_discontinue_date scan_modifier ispurchasable purchase_discontinue_date issellable sales_discontinue_dat
>..>e audit_flag'),(<'ItemSales'),(<;._1 ' day_date_0 calendar_key_0 organization_key product_key promotion_k
>..>ey shift_key etl_source_system_key extended_cost extended_purchase_rebate extended_sales_rebate extended_s
>..>ales purchase_rebate sales_rebate pack_size qty_sold inv_unit_qty ent_source_hdr_key ent_source_dtl_key ca
>..>lendar_key_1 day_date_1 day_of_week_id day_of_week holiday type_of_day calendar_month_no calendar_month_na
>..>me calendar_qtr_no calendar_qtr_desc calendar_year fiscal_week fiscal_period_no fiscal_period_desc fiscal_
>..>year ts'),(<'OrganizationView'),(<;._1 ' organization_key location_id location_desc last_tank_reading_dat
>..>e first_transaction_date last_remodel_date closing_date gps_latitude gps_longitude gps_address1 gps_addres
>..>s2 gps_city gps_state gps_zip corp_site business_entity_desc business_entity_id wholesale_marketing_desc w
>..>holesale_marketing_id wholesale_transportation_desc wholesale_transportation_id comparable_sites_at_distri
>..>ct_level_desc comparable_sites_at_district_level_id consolidated_oh_locations_desc consolidated_oh_locatio
>..>ns_id pwe_site_comparable_or_noncomparable_desc pwe_site_comparable_or_noncomparable_id comparable_by_stat
>..>e_desc comparable_by_state_id corporate_desc corporate_id company_desc company_id district_desc district_i
>..>d site_desc site_id construction_site_comparable_or_noncomp_desc construction_site_comparable_or_noncomp_i
>..>d paper_work_deployment_desc paper_work_deployment_id comp_non_comp_ci_desc comp_non_comp_ci_id state_ci_d
>..>esc state_ci_id area_ci_desc area_ci_id group_ci_desc group_ci_id payroll_group_ci_desc payroll_group_ci_i
>..>d company_ci_desc company_ci_id pb_corporate_desc pb_corporate_id pb_zone_desc pb_zone_id pb_division_desc
>..> pb_division_id pb_fountain_desc pb_fountain_id pb_budweiser_desc pb_budweiser_id pb_group_desc pb_group_i
>..>d pb_soda_desc pb_soda_id pb_tobacco_desc pb_tobacco_id pb_truckstops_desc pb_truckstops_id pb_coors_desc
```

```
>..>pb_coors_id pb_miller_desc pb_miller_id pb_chew_desc pb_chew_id pb_icee_desc pb_icee_id pb_candy_desc pb_c
>..>andy_id pb_coffee_desc pb_coffee_id pb_dairy_desc pb_dairy_id pb_tax2_btldp_desc pb_tax2_btldp_id pb_fuel_
>..>brand_desc pb_fuel_brand_id jackson_jet_center_llc_3_desc jackson_jet_center_llc_3_id breakout_between_fbo
>..>_flight_department_desc breakout_between_fbo_flight_department_id trucks_by_state_whlsl_desc trucks_by_sta
>..>te_whlsl_id desktop_deployment_desc desktop_deployment_id region_desc region_id rebate_manager_coke_zones_
>..>desc rebate_manager_coke_zones_id pb_alcohol_desc pb_alcohol_id store_cash_fund_desc store_cash_fund_id at
>..>m_refill_fund_desc atm_refill_fund_id car_wash_1_desc car_wash_1_id car_wash_2_desc car_wash_2_id car_wash
>..>_3_desc car_wash_3_id car_wash_4_desc car_wash_4_id a_company_desc a_company_id a_specific_brand_desc a_sp
>..>ecific_brand_id a_site_comparable_or_noncomparable_desc a_site_comparable_or_noncomparable_id a_maintenanc
>..>e_tech_region_desc a_maintenance_tech_region_id a_state_retail_setup_only_desc a_state_retail_setup_only_i
>..>d a_aquisition_group_desc a_aquisition_group_id a_pacwest_regions_desc a_pacwest_regions_id pb_bananas_des
>..>c pb_bananas_id truck_by_city_whlsl_desc truck_by_city_whlsl_id sites_now_in_enterprise_desc sites_now_in_
>..>enterprise_id weekly_reporting_group_desc weekly_reporting_group_id version_of_epos_software_system_desc v
>..>ersion_of_epos_software_system_id pb_egg_desc pb_egg_id rpt_weekly_category_id rpt_weekly_category_desc st
>..>ore_deposit_bank_name_id store_deposit_bank_name_desc rm_sites_for_fixed_rebates_desc rm_sites_for_fixed_r
>..>ebates_id pb_convection_id pb_convection_desc lottery_commission_rate_id lottery_commission_rate_desc pb_m
>..>ix_match_desc pb_mix_match_id timezone_id site_id_formatted rm_sites_id rm_sites_desc franchise_level_desc
>..> franchise_level_id site_distributor_id a_ytd_site_comparable_or_noncomparable_id a_ytd_site_comparable_or
>..>_noncomparable_desc'),(<'ProductView'),<<;._1 ' product_key upc_sell_unit_desc upc sell_unit_desc sell_uni
>..>t_qty item_id item_desc sub_category_id sub_category_desc category_id category_desc department_id departme
>..>nt_desc size_desc item_type grp_level_4_id grp_level_4_desc alt_grp_level_1_id alt_grp_level_1_desc alt_gr
>..>p_level_2_id alt_grp_level_2_desc alt_grp_level_3_id alt_grp_level_3_desc create_date ent_item_number upc_
>..>discontinue_date scan_modifier ispurchasable purchase_discontinue_date issellable sales_discontinue_date a
>..>udit_flag washington_tax status scan_data scan_based report_ff_combined pos_weight_uom_basis pos_verifone_
>..>item_type_sub_code pos_verifone_item_type_code pos_use_department_tax pos_upc_type pos_tare_method pos_spe
>..>edkey_number pos_scaleable_item pos_scale_unit_of_measure pos_scale_tare_method pos_sales_unit_of_measure
>..>pos_retalix_item_type_sub_code pos_radiant_item_type_sub_code pos_radiant_item_type_code pos_prompt_for_pr
```

```
>..>ice pos_promotion_reason pos_prohibit_item_returns pos_product_type pos_pinnacle_palm_department_type_code
>..> pos_passport_naxml_3_4_item_type_sub_code pos_passport_naxml_3_4_item_type_code pos_passport_department_t
>..>ype_code pos_nixdorf_vynamic_item_type_code pos_ngrp_item_type_code pos_naxml_scale_unit_of_measure pos_na
>..>xml_combo_mix_match_priority pos_naxml_3_6_payment_systems_product_code pos_naxml_3_6_link_code_type pos_n
>..>axml_3_6_item_type_sub_code pos_naxml_3_6_item_type_code pos_naxml_3_5_link_code_type pos_naxml_3_5_item_t
>..>ype_code pos_naxml_3_4_link_code_type pos_naxml_3_4_item_type_code pos_naxml_3_3_item_type_code pos_mm_sou
>..>rce pos_mm_bt9000_tender_restriction pos_mm_bt9000_card_swipe_type_15_char pos_loyalty_points_multiplier_t
>..>ype pos_link_code_type pos_incomm_api_type pos_host_product_code pos_gas_pos_item_type_code pos_fractional
>..>_quantity_allowed pos_flexible_spending_type pos_ebt_allowed_avalara_or pos_ebt_allowed_avalara pos_ebt_al
>..>lowed pos_deal_group_ctp_award_program pos_credit_network_product_code_3_char pos_coupon_tender_type pos_c
>..>onexxus_product_code pos_car_wash_type pos_car_wash_position_on_crind_screen pos_car_wash_controller_code_
>..>12_char pos_buypass_numeric_product_codes_2010_10_25 pos_buypass_alpha_product_codes_2010_10_25 pos_bt9000
>..>_coupon_item_number pos_blue_law_1_applies pos_allow_item_returns pos_age_verification_method pdisa_sectio
>..>n_audit_count_method manufacturer kpi_traits key_performance_indicators item_tax_group idaho_sales_tax_nob
>..>taxable_items brand'


SwiftNoGuidMsg=: 'NO-GUID-SET'

SwiftOKRc=: 0

SwiftOkLogFile=: 'aa_OK_SwiftIQ.txt'

SwiftTableSfx=: <;._1 ' Calendar FuelProduct ItemSales OrganizationView ProductView'

SwiftTsvDir=: '\\jfsdev04\Shares\SwiftIQPreprocess\SwiftTsv\'
```

```
SwiftZipArchiveDir=: '\\jfscorpftp01\FTPSites\SwiftIQ\Outbound\Archive\'

SwiftZipCsvDir=: '\\jfsdev04\Shares\SwiftIQPreprocess\SwiftZipCsv\'

NB. tab character
TAB=: 9{a.

ireadapply=: 1 : 0

NB.*ireadapply v-- apply verb (u) to n byte line blocks of static file.
NB.
NB. adv: u ireadapply (clFileIn ; clFileOut ; clDel ; iaBlockSize ;< uuData)
NB.
NB.   fi=. winpathsep ;1 dir SwiftZipCsvDir,'ItemSales-*.csv'
NB.   fo=. SwiftTsvDir,'land_ItemSales.txt'
NB.   smoutput@:(>@{. ; ($&.>)@:}.) ireadapply fi;fo;CRLF;20000000;<''

NB. file in, file out, line delimiter, block size, (u) verb data
'fi fo d k ud'=. y

p=. 0          NB. file pointer
c=. 0          NB. block count
s=. fsize fi   NB. file bytes
k=. k<.s       NB. first block size
NB.debug. b=. i.0  NB. block sizes (chk)

while. p < s do.
```

```
  'iread error' assert -. _1 -: r=. (1!:11 :: _1:) fi;p,k
  c=. >:c  NB. block count
  NB. complete lines
  if. 0 = #l=. d beforelaststr r do.
    NB. final shard
    NB.debug. b=. b,#r
    u c;1;d;fo;r;<ud break.
  end.
  p=. p + #l       NB. inc file pointer
  k=. k <. s - p   NB. next block size
  NB.debug. b=. b,#l      NB. block sizes list
  NB. block number, shard, delimiter, file out, line bytes, (u) data
  u c;0;d;fo;l;<ud
end.

NB.debug. 'byte mismatch' assert s = +/b
c  NB. blocks processed
)

SwiftErrorLog=: 3 : 0

NB.*SwiftErrorLog v-- write/rename Swift error log file.
NB.
NB. monad:  paRc =. SwiftErrorLog (iaRc ; clMsg)

ltx=. '' [ 'rc msg'=. y
if. fexist okf=. SwiftTsvDir,SwiftOkLogFile do. ferase okf [ ltx=. read okf end.
smoutput ltx=. ltx,LF,LogMsgPfx,(nowyyyymmddhrnnss 0),LF,msg
```

```
rc [ (toHOST ltx) write SwiftTsvDir,SwiftErrorLogFile
)

SwiftLog=: 3 : 0

NB.*SwiftLog v-- append message to SwiftIQ log.
NB.
NB. monad:  paRc =. SwiftLog clMsg

smoutput y
ltx=. LF,LogMsgPfx,(nowyyyymmddhrnnss 0),LF,y
SwiftOKRc [ (toHOST ltx) fappend SwiftTsvDir,SwiftOkLogFile
)

NB. retains string after first occurrence of (x)
afterstr=: ] }.~ #@[ + 1&(i.~)@([ E. ])

NB. trims all leading and trailing white space
allwhitetrim=: ] #~ [: -. [: (*./\. +. *./\) ] e. (9 10 13 32{a.)"_

appendswiftcsvs=: 3 : 0

NB.*appendswiftcsvs v-- append csvs of the same  type to a single
NB. TAB delimited file.
NB.
NB. monad:  iaRc =. appendswiftcsvs bl
```

```
'fo cols files'=. y
fo=. SwiftTsvDir,fo,'.txt'
SWIFTAPPENDCNT_base_=:  0

for_file. files do.
  smoutput fi=. ;file
  NB. current header - csv columns vary
  chdr=. swiftheader fi
  'current csv has unknown column(s)' assert chdr e. cols
  NB. moves current columns to landing positions
  NB. assumes all current csv rows match header positions
  pmc=. (#cols),cols i. chdr
  (smoutput@;)@:apptabFrswiftIQcsv ireadapply fi;fo;CRLF;SwiftIreadBytes;<pmc
end.

SwiftLog (":SWIFTAPPENDCNT_base_),'|',fo
)

apptabFrswiftIQcsv=: 3 : 0

NB.*apptabFrswiftIQcsv v-- (ireadapply) verb append TAB delimited blocks.
NB.
NB. monad:  apptabFrswiftIQcsv (iaCnt ; paShard ; clDel ; clFile ; clBytes ;< uuData)

NB. block number, shard, delimiter, file, line bytes, (u) data
'c s d fo l ud'=. y

NB. if. s do. s shard l end.
```

```
NB. shard is eof
if. 0 = #l -. CRLF do. c;0 return. end.

NB. replace ',' in quotes with ';'
t=. ',;' repdqchars (2 * CRLF -: 2 {. l) }. l

NB. parse after removing quotes and TABS
if. #t=. allwhitetrim&.> ',' parsecd t -. '"',TAB do.

  NB. move current columns to landing positions
  t=. t (<a:;}.ud)} a: $~ (#t),{.ud

  NB. first block has header - already written
  if. #t=. (c=1) }. t do.
    SWIFTAPPENDCNT_base_=:  SWIFTAPPENDCNT_base_ + #t
    (toHOST fmttd t) fappend fo
  end.

end.

c;$t
)

NB. signal with optional message
assert=: 0 0"_ $ 13!:8^:((0: e. ])`(12"_))

NB. retains string (y) before last occurrence of (x)
```

```
beforelaststr=: ] {.~ 1&(i:~)@([ E. ])

NB. retains string before first occurrence of (x)
beforestr=: ] {.~ 1&(i.~)@([ E. ])

NB. boxes open nouns
boxopen=: <^:(L. = 0:)

changestr=: 4 : 0

NB.*changestr v-- replaces substrings - see long documentation.
NB.
NB. dyad:  clReps changestr cl
NB.
NB.   NB. first character delimits replacements
NB.   '/change/becomes/me/ehh' changestr 'blah blah ...'

pairs=. 2 {."(1) _2 [\ <;._1 x      NB. change table
cnt=._1 [ lim=. # pairs
while. lim > cnt=.>:cnt do.         NB. process each change pair
  't c'=. cnt { pairs               NB. /target/change
  if. +./b=. t E. y do.             NB. next if no target
    r=. I. b                        NB. target starts
    'l q'=. #&> cnt { pairs         NB. lengths
    p=. r + 0,+/\(<:# r)$ d=. q - l NB. change starts
    s=. * d                         NB. reduce < and > to =
    if. s = _1 do.
```

```
      b=. 1 #~ # b
      b=. ((l * # r)$ 1 0 #~ q,l-q) (,r +/ i. l)} b
      y=. b # y
      if. q = 0 do. continue. end.  NB. next for deletions
    elseif. s = 1 do.
      y=. y #~ >: d r} b            NB. first target char replicated
    end.
    y=.(c $~ q *# r) (,p +/i. q)} y  NB. insert replacements
  end.
end. y                              NB. altered string
)

charsub=: 4 : 0

NB.*charsub v-- single character pair replacements.
NB.
NB. dyad:  clPairs charsub cu
NB.
NB.    '-_$ ' charsub '$123 -456 -789'

'f t'=. ((#x)$0 1)<@,&a./.x
t {~ f i. y
)

checkswiftfiles=: 3 : 0

NB.*checkswiftfiles v-- checks working directory SwiftIQ csvs and
```

```
NB. zip files.
NB.
NB. monad:  paRC =. checkswiftfiles clFiles
NB. dyad:  ?? checkswiftfiles ??

1
)

clearswiftdirs=: 3 : 0

NB.*clearswiftdirs v-- clears working SwiftIQ directories.
NB.
NB. monad:  iaRc =. clearswiftdirs uuIgnore

NB. clear only working files !(*)=. dir
files=.       1 dir SwiftZipCsvDir,'*.csv'
files=. files,1 dir SwiftZipCsvDir,'*.zip'
files=. files,1 dir SwiftTsvDir,'*.txt'
NB. might need to rescan incase file permissions get changed
SwiftOKRc [ ferase files
)

cntswiftrows=: 3 : 0

NB.*cntswiftrows v-- SwiftIQ differance/reporting/stage row counts.
NB.
NB. NOTE: after a successful update the values should be '0 X X',
```

```
NB. i.e. the reporting and staging item  counts should  match and
NB. the difference should be zero.
NB.
NB. monad:  clCnts =. cntswiftrows uuIgnore

'rcnt rmax'=. ,RepCntSwiftIQ_sql fetc ODBCSWIFTREPORT
'scnt sdif'=. ,(('/{~{~RKEY~}~}/',":rmax) changestr StgCntSwiftIQ_sql) fetc ODBCSWIFT
;(":&.> sdif;rcnt;scnt) ,&.> ' ' NB. cl for logging
)


copyswiftzip=: 3 : 0

NB.*copyswiftzip v-- copies SwiftIQ zip to working directory.
NB.
NB. monad:  clWorkZip =. copyswiftzip clZip
NB.
NB.   copyswiftzip ;0{swiftZips 0

NB. require 'task' !(*)=. shell
cmd=. 'copy ',(enquote y),' ',enquote of=. SwiftZipCsvDir,justfileext y
of [ shell cmd
)


dailyswift=: 3 : 0

NB.*dailyswift v-- daily SwiftIQ zip loading and maintenance tasks.
NB.
```

```
NB. monad:  (ilRc ; clGuid) =. dailyswift uuIgnore
NB. dyad:  (ilRc ; clGuid) =. iaDayCode dailyswift uuIgnore
NB.
NB.   _1 dailyswift 0  NB. update stage only
NB.   dailyswift~ _1
NB.
NB.   NB. full table reload
NB.   42 dailyswift 0

(weekday 3 {. 6!:0 '') dailyswift y
:
NB. stage latest zip
'rc tg'=. procswiftzip ;0{swiftZips 0
if. 0~:0{rc do. rc;tg return. end.

if. _1=x do.
  rc=. SwiftLog 'only updating staging'
else.
  rcnts=. cntswiftrows 0  NB. report/stage item counts

  NB. update reporting
  rc=. logswiftReportBegin tg;rcnts
  if. 0=x do.
    SwiftLog 'rowcnts::pidx: ',rcnts
    SwiftLog 'reindexing - partial table update'
    rc=. runswiftssis 'swiftreindex.bat'
    rc=. runswiftssis 'swiftreport.bat'
```

```
  elseif. 42=x do.
    NB. full updates are best babysat - trigger with
    NB. code 42 - life the universe and everything
    SwiftLog 'rowcnts::full: ',rcnts
    SwiftLog 'full table refresh'
    rc=. runswiftssis 'swiftreportfull.bat'

  elseif.do.
    SwiftLog 'rowcnts:: ',rcnts
    SwiftLog 'partial table update'
    rc=. runswiftssis 'swiftreport.bat'
  end.
  rc=. logswiftReportEnd rc;tg
end.
)

NB. enclose in " quotes - surrounding blanks removed
enquote=: '"' , '"' ,~ ] #~ [: -. [: (*./\. +. *./\) ' '&=

NB. append to file (UTF8 file names) returns #bytes if successful _1 otherwise
fappend=: ([: , []) (#@[ [ 1!:3) ::(_1:) [: fboxname ]

NB. boxes UTF8 names
fboxname=: ([: < 8 u: >) ::]

NB. erase files - cl | blcl of path file names
ferase=: 1!:55 ::(_1:)@(fboxname&>)@boxopen
```

```
fetc=: 4 : 0

NB.*fetc v-- ddfet sql data from dsn.
NB.
NB. dyad:  bt =. clSql fetc iaCh
NB.
NB.   'select top 10 * from Daily_Etl_Log_New order by Logdatetime desc' fetc 'dsn=emcs_staging'

NB. require 'dd' !(*)=. ddsel ddfet ddcon dddis
'cannot connect db'  assert 0<ch=. ddcon y
'cannot select data' assert 0<s=. x ddsel ch
'cannot fetch data'  assert -. _1-:d=. ddfet s,_1
d [ dddis ch
)

NB. 1 if file exists 0 otherwise
fexist=: 1:@(1!:4) ::0:@(fboxname&>)@boxopen

NB. format tables as TAB delimited LF terminated text - see long document
fmttd=: [: (] , ((10{a.)"_ = {:) }. (10{a.)"_) [: }.@(,@(1&(,"1)@(-.@(*./\."1@(=&' '@])))) # ,@((10{a.)&(,"
>..>1)@])) [: }."1 [: ;"1 (9{a.)&,@":&.>


NB. size of file in bytes
fsize=: 1!:4 ::(_1:)@(fboxname&>)@boxopen

fullheaderscan=: 3 : 0
```

```
NB.*fullheaderscan v--  scans all  the  SwiftIQ  zips and  builds
NB. complete table headers.
NB.
NB. monad:  bt =. fullheaderscan uuIgnore
NB.
NB.   SwiftLandTableHeaders=: fullheaderscan 0
NB.
NB. dyad: bt =. clZipFile fullheaderscan uuIgnore
NB.
NB.   'JacksonsFiles0803202063717.zip' fullheaderscan 0

'' fullheaderscan y
:
zips=. x swiftZips 0

NB. all table column names
acns=. SwiftTableSfx ,. a:

for_zip. zips do.
  loadswiftcsvs ;zip [ clearswiftdirs 0
  NB. j profile !(*)=. dir
  if. #csvs=. 1 dir SwiftZipCsvDir,'*.csv' do.
    dtns=. tablenamesfrcsvs csvs
    'unexpected table name(s)' assert dtns e. SwiftTableSfx
    hdns=. swiftheader&.> csvs
    ix=. SwiftTableSfx i. dtns
    NB. table column name union
```

```
    acns=. (~.&.>((<ix;1){acns),&.> hdns) (<ix;1)} acns
  else.
    smoutput 'no csv(s) or corrupt zip: ', , ;zip
  end.
  NB.debug. if. zip_index > 3 do. break. end.
end.

acns
)

guids=: 3 : 0

NB.*guids v-- create guids as 16 byte strings on supported J systems.
NB.
NB. This verb taken from ~addons/general/misc/guids.ijs returns guids
NB. on Windows, Linux and Mac systems.
NB.
NB. monad:  guids zl | ilShape
NB.
NB.   guids ''    NB. create guid as a 16-byte character string
NB.   guids $0
NB.   guids 3 4   NB. create 3x4 array of 16-byte strings

NB. standard J profile !(*)=. IFWIN UNAME
if. IFWIN do.
  cmd=. 'ole32 CoCreateGuid i *c'
else.
  cmd=. ((UNAME-:'Darwin'){::'libuuid.so.1';'libSystem.B.dylib'),' uuid_generate n *c'
```

```
end.
>{:"1 cmd 15!:0"1 0 <"1 (y,16)$' '
)

NB. hexidecimal 32 byte guids: guidsh 7 2
guidsh=: [: ,@hfd"1 a. i. guids

NB. hex from decimal: hfd 5078
hfd=: 16&#.@('0123456789ABCDEF'&i.)^:_1

NB. file name and extension from fully qualified file
justfileext=: ] #~ [: -. [: +./\. '\'&=

loadswiftcsvs=: 3 : 0

NB.*loadswiftcsvs v-- copies zip to SwiftIQ working directory and
NB. extracts csvs.
NB.
NB. monad:  blclFiles =. loadswiftcsvs clZipFile
NB.
NB.   loadswiftcsvs ;0{swiftZips 0   NB. latest file
NB.   loadswiftcsvs ;14{swiftZips 0  NB. about two weeks back
NB.
NB.   NB. contains very large csvs - good test cases
NB.   loadswiftcsvs SwiftZipArchiveDir,'JacksonsFiles08062020112547.zip'
NB.   loadswiftcsvs SwiftZipArchiveDir,'JacksonsFiles0801202072250.zip'
```

```
NB. j profile !(*)=. dir
unzipswift copyswiftzip y
csvs=. winpathsep&.> 1 dir SwiftZipCsvDir,'*.csv'
NB. zip precedes csvs
(winpathsep&.> 1 dir SwiftZipCsvDir,'*.zip'),csvs
)

loadswifttabs=: 3 : 0

NB.*loadswifttabs v-- convert SwiftIQ csvs to TAB delimited table files.
NB. files.
NB.
NB. monad:  iaRc =. loadswifttabs blclCsvs
NB.
NB.   csvs=. loadswiftcsvs ;0{swiftZips 0   NB. latest file
NB.   csvs=. loadswiftcsvs ;_1{swiftZips 0  NB. first file
NB.   loadswifttabs csvs

NB. first file is zip
if. 0 e. $csvs=. }.y do.
  SwiftErrorLog 205;'err::zip empty or corrupt - nothing loaded: ',;{.y return.
end.

NB. fetch expected landing tables/columns from SQL
tcols=. swiftlandcols 0 [ lpfx=. 'land_'

NB. current csv prefixes should match landing tables
fpfx=. (('-'&beforestr)@justfileext&.> csvs) -.&.> '_'
```

```
land=. (<lpfx) ,&.> fpfx
'unexpected csv file name prefix(s)' assert land e. 0 {"1 tcols

NB. partition csvs by prefix
upix=. ~.pix=. ((#lpfx) }.&.> 0 {"1 tcols) i. fpfx
csvs=. (upix{tcols) ,. pix </. csvs

txts=. writeswiftHeaders tcols
for_csv. csvs do. rc=. appendswiftcsvs csv end.
runswiftssis 'swiftstage.bat'
)

logswiftBegin=: 3 : 0

NB.*logswiftBegin v-- logs start of SwiftIQ zip load.
NB.
NB. monad:  clGuid =. logswiftBegin uuIgnore

NB. require 'dd' !(*)=. ddcon ddsql dddis
'unable to connect swift staging' assert 0<ch=. ddcon ODBCSWIFT
tg=. ,guidsh 1
sql=. 'insert into log_EtlTasks (TaskRC,TaskID,TaskItem) values '

NB. task starts set TaskRC = 0 - ok code
sql=. sql,'(0,',(quote tg),',',(quote '::BEGIN>> swift csv zip load'),')'
tg;rc [ dddis ch [ rc=. sql ddsql ch
)
```

```
logswiftEnd=: 3 : 0

NB.*logswiftEnd v-- logs end of SwiftIQ csv zip load.
NB.
NB. monad:  (ilRc ; clGuid) =. logswiftEnd (iaRc ; clGuid ; clZipFile)
NB.
NB.    NB. test case
NB.   logswiftEnd 200;(,guidsh 1);'zipfiletest.zip'

NB. require 'dd' !(*)=. ddcon ddsql dddis
'trc tg zip'=. y

tit=. '::END>> swift csv zip load'
'unable to connect swift staging' assert 0<ch=. ddcon ODBCSWIFT

NB. log_LoadedFiles row
sql=. 'insert into log_LoadedFiles (TaskID,TaskItem,LoadZipName,LoadMessage) values '
svl=. quote tg
svl=. svl,',',quote tit
svl=. svl,',',quote justfileext zip

NB. collect start/end log text - fit into db column
ltx=. allwhitetrim ;(read :: ''"_) &.> (<SwiftTsvDir) ,&.> SwiftOkLogFile;SwiftErrorLogFile
if. LogTextLim < #ltx do. ltx=. ,(1 _1 * <.@-:LogTextLim) {."0 1 ltx end.
svl=. svl,',',quote ltx
sql=. sql,'(',svl,')'
rc=. sql ddsql ch
```

```
NB. log_EtlTasks log row - return codes >: 0
sql=. 'insert into log_EtlTasks (TaskRC,TaskID,TaskItem) values '
sql=. sql,'(',(":|trc),',',(quote tg),',',(quote tit),')'
(trc,rc);tg [ dddis ch [ rc=. rc,sql ddsql ch
)

logswiftLast=: 3 : 0

NB.*logswiftLast v-- last logged SwiftIQ message text.
NB.
NB. monad:  blcl =. logswiftLast uuIgnore
NB.
NB.   ;{:logswiftLast 0

,'select top(1) TaskID, loadmessage from log_LoadedFiles order by LoadUtcDatetime desc' fetc ODBCSWIFT
)

logswiftReportBegin=: 3 : 0

NB.*logswiftReportBegin v-- logs start of reporting database update.
NB.
NB. monad:  iaRc =. logswiftReportBegin (clStageGuid ; clRowCnts)
NB.
NB.   logswiftReportBegin (tg=: ,guidsh 1);cntswiftrows 0

'tg rcnts'=. y
```

```
NB. require 'dd' !(*)=. ddcon ddsql dddis
'unable to connect swift report' assert 0<ch=. ddcon ODBCSWIFTREPORT
sql=. 'insert into log_EtlTasks (TaskRC,TaskID,TaskItem) values '

NB. task starts set TaskRC = 0 - ok code
sql=. sql,'(0,',(quote tg),',',(quote '::BEGIN>> swift report update:',rcnts),')'
rc [ dddis ch [ rc=. sql ddsql ch
)

logswiftReportEnd=: 3 : 0

NB.*llogswiftReportEndv-- logs end of reporting database update.
NB.
NB. monad:  (ilRc ; clGuid) =. logswiftEnd (iaRc ; clGuid)
NB.
NB.    NB. test case
NB.   logswiftReportEnd 200;,guidsh 1

NB. require 'dd' !(*)=. ddcon ddsql dddis
'trc tg'=. y

tit=. '::END>> swift report update'
'unable to connect swift report' assert 0<ch=. ddcon ODBCSWIFTREPORT

NB. log_EtlTasks log row - return codes >: 0
sql=. 'insert into log_EtlTasks (TaskRC,TaskID,TaskItem) values '
sql=. sql,'(',(":|trc),',',(quote tg),',',(quote tit),')'
```

```
(trc,rc);tg [ dddis ch [ rc=. sql ddsql ch
)

nowyyyymmddhrnnss=: 3 : 0

NB.*nowyyyymmddhrnnss v-- now in yyyymmddhrnnss cl format.
NB.
NB. monad:  clNow =. nowyyyymmddhrnnss uuIgnore

(":{.t) , ,'r<0>2.0'&(8!:2) }. t=. <.6!:0 ''
)

NB. parse (x) delimited table text - see long document
parsecd=: [: <;._2&> [ ,&.>~ [: <;._2 [: (] , ((10{a.)"_ = {:) }. (10{a.)"_) (13{a.) -.~ ]

procqueuezips=: 3 : 0

NB.*procqueuezips v-- processes queued zips from file.
NB.
NB. monad:  btRcGuid =. procqueuezips clQueueFile
NB.
NB.   procqueuezips SwiftAuxDir,'swift_zip_small.txt'
NB.   procqueuezips SwiftAuxDir,'swift_zip_normal.txt'
NB.   procqueuezips SwiftAuxDir,'swift_zip_large.txt'
NB.   procqueuezips SwiftAuxDir,'swift_zip_first_month.txt'
NB.
NB.   NB. n=5 random zips
```

```
NB.    ztxt=: toHOST ;(([] { ~ 5 ? #) swiftZips 0) ,&.> LF
NB.    ztxt write SwiftAuxDir,'swift_zip_random.txt'
NB.    procqueuezips SwiftAuxDir,'swift_zip_random.txt'
NB.
NB. dyad:  btRcGuid =. paReindex procqueuezips clQueueFile
NB.
NB.    1 procqueuezips SwiftAuxDir,'swiftN.txt'

0 procqueuezips y NB. do not reindex (default)
:
if. 1-:x do.
  if. -.SwiftOKRc-:rc=. runswiftssis 'swiftreindex.bat' do.
    ,:rc;SwiftNoGuidMsg return.
  end.
end.

NB. get list of queued zips
zips=. 0$a: [ txt=. '' [ clearswiftdirs 0
if. fexist y do. txt=. (read :: ''"_) y end.
if. #txt do.
  zips=. <;._2(tlf txt) -. CR
  if. 0 e. zips e. swiftZips 0 do.
    ,:SwiftNoGuidMsg ;~ SwiftErrorLog 130;'err:invalid zip(s) on file queue' return.
  end.
else.
  ,:SwiftNoGuidMsg ;~ SwiftErrorLog 120;'err:zip queue file missing or empty' return.
end.
```

```
rc=. 0 2$a: [ zcnt=. #zips
for_zip. zips do.
  smoutput (>:zip_index),zcnt
  rc=. rc,(procswiftzip :: topswiftTrap) ;zip
end.

rc
)

procswiftzip=: 3 : 0

NB.*procswiftzip v-- process a SwiftIQ zip.
NB.
NB. Extracts SwiftIQ csvs from zip, converts them to standard TAB
NB. delimited table files, and then loads the TAB delimited files
NB. to SwiftIQ_staging by running an SSIS package.
NB.
NB. monad:  (ilRc ; clGuid) =. procswiftzip clZipFile
NB.
NB.   procswiftzip ;0{swiftZips 0  NB. latest file
NB.
NB.   NB. never loaded in load order
NB.   procswiftzip&> SwiftZipArchiveDir&,&.> showswiftUnload 0
NB.
NB.   NB. last in chronological order
NB.   procswiftzip&> |. 10 {. swiftZips 0
NB.
```

```
NB.   NB. random archive zip
NB.   procswiftzip ;(] { ~ [: ? #) swiftZips 0
NB.
NB.   NB. error trapped
NB.   (procswiftzip :: topswiftTrap) ;0{swiftZips 0

'tg rc'=. logswiftBegin 0

NB. set guid for error trapping
SWIFTTASKGUID_base_=:  tg
SwiftLog 'started::',tg [ clearswiftdirs 0

if. -.fexist y do.
  rc=. SwiftErrorLog 310;'err::zip file (y) does not exist - nothing loaded'
  logswiftEnd rc;tg;'' return.
end.

SwiftLog 'zip::',y

NB. check for prior zip load - default is to check
if. CheckPriorSwiftZip do.
  if. +./(<justfileext y) e. swiftLoaded '=0' do.
    NB. indicates problems with upstream tasks
    rc=. SwiftErrorLog 305;'err::zip has been previously loaded : ',y
    logswiftEnd rc;tg;y return.
  end.
end.
```

```
files=. loadswiftcsvs y
SwiftLog 'csvs::',;('|'&,)@:justfileext&.> }.files
if. checkswiftfiles files do.
  rc=. loadswifttabs files
else.
  rc=. SwiftErrorLog 100;'err::invalid working files'
end.
logswiftEnd rc;tg;y
)

NB. file sizes of SwiftIQ queues: queueswiftSizes SwiftAuxDir,'swift2.txt'
queueswiftSizes=: [: (([: <"0 [: >: [: i. #) ,. ] ,.~ [: <"0 fsize) [: <;._2 [: tlf (13{a.) -.~ read

queueswiftZips=: 3 : 0

NB.*queueswiftZips v-- write (y) line file of queued zips.
NB.
NB. Files   generated   by   this   verb   are   processed   with
NB. (procqueuezips).
NB.
NB. monad:  (ilN ; clFile) =. queueswiftZips ia
NB. dyad:  (ilN ; clFile) =. il queueswiftZips ia

_50 1 queueswiftZips y
:
file=. SwiftAuxDir,'swift',(":y),'.txt'
```

```
nzcnt=. #zips=. (0{x) <\ |. swiftZips 0
'selection out of range' assert y < nzcnt
zips=. ;y { zips

if. 1-:1{x do.
  zlod=. swiftLoaded '=0'
  b=. (justfileext&.> zips) e. zlod
  smoutput (":+/b),' previously loaded - removed'
  zips=. zips #~ -.b
end.

(y,nzcnt);file [ (toHOST ;zips ,&.> LF) write file
)


NB. quotes character lists for execution
quote=: ''''&,@(,&'''')@(#~ >:@(=&''''))

NB. reads a file as a list of bytes
read=: 1!:1&(]`<@.(32&>@(3!:0)))

reb=: 3 : 0

NB.*reb v-- removes redundant blanks - leading, trailing multiple
NB.
NB. monad:  reb cl
NB. dyad:  ua reb ul
```

```
' ' reb y
:
y=. x , y
b=. x = y
}.(b*: 1|.b)#y
)

repdqchars=: 4 : 0

NB.*repdqchars v-- replace double quoted (0{x) characters with (1{x).
NB.
NB. dyad:  cl =. clPair repdqchars cl
NB.
NB.   s=. '"go ahead, replace","quoted commas,,,,"'
NB.   ',;' repdqchars s
NB.   s -: ',,' repdqchars s

if. 2 <: #y do.
  msg=. 'unbalanced quotes'
  msg assert 0 = 2 | +/b=. '"'=y

  NB. mask of quoted characters
  q=. +/\ _1 (1 { |: _2 ]\ I. b)} b
  msg assert 0 1 -: (<./ , >./) q

  NB. replace quoted (0{x) with (1{x)
  (1{x) (I. q #^:_1 (0{x) = q#y) } y
else.
```

```
  y
end.
)

runswiftssis=: 3 : 0

NB.*runswiftssis v-- runs SwiftIQ SSIS batch scripts.
NB.
NB. monad:  iaRc =. runswiftssis clScript
NB.
NB.   runswiftssis 'swiftstage.bat'    NB. process TAB files
NB.   runswiftssis 'swiftreport.bat'   NB. reload wh tables
NB.   runswiftssis 'swiftreindex.bat'  NB. reindex tables

if. -.RUNSSISFLAG do.
  SwiftLog 'SSIS package execution is off - not run: ',y return.
end.

if. fexist scr=. SwiftCodeDir,y do.

  SwiftLog 'starting SSIS package: ',y
  NB. require 'task' !(*)=. shell
  NB. run ssis package script
  ctxt=. shell scr
  if. DTExecOKMsg +./@E. ctxt do.
    SwiftLog DTExecOKMsg
  else.
    SwiftErrorLog 105;('err::ssis script (',y,') returned error(s)'),LF,ctxt
```

```
  end.

else.
  SwiftErrorLog 205;'err:: script (',scr,') does not exist'
end.
)

sfxnonunique=: 4 : 0

NB.*sfxnonunique  v-- attach distinct  suffixes  to nonunique  cl
NB. items.
NB.
NB. dyad:  blcl =. cl sfxnonunique blcl
NB.
NB.   s=. ;:'i do what i do because i am what i am'
NB.   t=. ;:'all unique no suffixes'
NB.
NB.   '_' sfxnonunique s
NB.   's-' sfxnonunique s
NB.    t -: '-' sfxnonunique t

NB. sorted nonunique positions
b=. y e. y #~ -. ~: y
t=. (<"0@I. b) ,: b # y
t=. (/: 1{t) {"1 t

NB. distinct suffixes on nonunique
t=. ((1{t) ,&.> ;<"1&.> (<x) ,"1&.> ":@,.@i.&.> <"0 #/.~ 1{t) 1} t
```

```
(1{t) (;0{t)} y
)

shard=: 4 : 0

NB.*shard v-- inspect any final (ireadapply) shard.
NB.
NB. dyad:  pa shard cl

smoutput (100 <. #y) {. y
)

showswiftUnload=: 3 : 0

NB.*showswiftUnload v-- list never loaded zips in load order.
NB.
NB. monad:  blclZips =. showswiftUnload uuIgnore
NB. dyad:  blclZips =. clFilter showswiftUnload uuIgnore
NB.
NB.   NB. zips that have not been loaded without errors
NB.   showswiftUnload~ '=0'

NB. default all loaded including error runs
'<1000' showswiftUnload y
:
|. (justfileext&.> swiftZips 0) -. swiftLoaded x
)
```

```
showswiftlog=: 3 : 0

NB.*showswiftlog v-- show SwiftIQ log entries.
NB.
NB. monad:  bt =. showswiftlog clFrg
NB.
NB.   showswiftlog '=0'     NB. successful runs
NB.   showswiftlog '<>0'    NB. failures
NB.   showswiftlog '=105'   NB. 105 failures
NB.   showswiftlog '<1000'  NB. all runs
NB.
NB. dyad: bt =. iaN showswiftlog clFrg
NB.
NB.   10 showswiftlog '<1000'  NB. last ten runs

0 showswiftlog y
:
sr=. '/{~{~TOPN~}~}/',((0<x)#' top ',":x),'/{~{~RC~}~}/',y
(sr changestr LogSwiftIQ_sql) fetc ODBCSWIFT
)

NB. session manager output
smoutput=: 0 0 $ 1!:2&2

swiftFail=: (777;'ERROR-NO-GUID')"_

swiftLandingColumns=: 3 : 0
```

```
NB.*swiftLandingColumns v-- current SwiftIQ staging table
NB. columns.
NB.
NB. monad:  btcl =. swiftLandingColumns uuIgnore
NB. dyad:  btcl =. clPfx swiftLandingColumns uuIgnore
NB.
NB.   'raw_' swiftLandingColumns 0
NB.   swiftLandingColumns~ 'stage_'

'land_' swiftLandingColumns y
:
NB. require 'dd' !(*)=. ddcon ddcol dddis
if. 0 > ch=. ddcon ODBCSWIFT do.
  smoutput 'cannot connect staging'
  0 2$a:
else.
  hd=. ;:'TABLE_NAME COLUMN_NAME'
  d=.  (x,'Calendar') ddcol ch
  tc=. }.((0{d) i. hd) {"1 d
  d=.  (x,'FuelProduct') ddcol ch
  tc=. tc , }.((0{d) i. hd) {"1 d
  d=.  (x,'ItemSales') ddcol ch
  tc=. tc , }.((0{d) i. hd) {"1 d
  d=.  (x,'OrganizationView') ddcol ch
  tc=. tc , }.((0{d) i. hd) {"1 d
  d=.  (x,'ProductView') ddcol ch
  tc=. tc , }.((0{d) i. hd) {"1 d
```

```
  hd , tc [ dddis ch
end.
)

NB. loaded swift zips with return code: swiftLoaded '<1000'
swiftLoaded=: 5 { "1 showswiftlog

swiftMissingColumns=: 3 : 0

NB.*swiftMissingColumns v-- list missing SwiftIQ table columns.
NB.
NB. monad:  bt =. swiftMissingColumns uuIgnore
NB. dyad:  bt =. clPfx swiftMissingColumns uuIgnore

'land_' swiftMissingColumns y
:
NB. current table columns
cc=. allwhitetrim&.> }.x swiftLandingColumns 0
cc=. ((#x) }.&.> 0 {"1 cc) ,. 1 {"1 cc

NB. union of scanned zips see (fullheaderscan)
sh=. SwiftLandTableHeaders
sh=. allwhitetrim&.> ;(<"0 [ 0 {"1 sh) ,.&.> 1 {"1 sh

(('Not in ',x,' Tables');<sh -. cc) ,: 'Not in file CSVs';<cc -. sh
)
```

```
swiftTrap=: 3 : 0

NB.*swiftTrap v-- executed by (topswiftTrap).
NB.
NB. monad:  iaRc =. swiftTrap uuIgnore

msg=. 'Jerr::',13!:12 ''
tg=. SwiftNoGuidMsg
if. 0 = (4!:0) <'SWIFTASKGUID_base_' do. tg=.SWIFTASKGUID_base_ end.
rc=. (SwiftErrorLog :: 777"_) 666;msg
(logswiftEnd :: swiftFail) rc;tg;msg
)

swiftZips=: 3 : 0

NB.*swiftZips v-- sorted SwiftIQ zip files.
NB.
NB. Returns a  list  of sorted  SwiftIQ  zip files. Order is from
NB. most recent to oldest based om  the mmddyyyyhhnnss time stamp
NB. embedded in the file name.
NB.
NB. monad:  blclFiles =. swiftZips uuIgnore
NB. dyad: blclFiles =. clZipFile swiftZips uuIgnore
NB.
NB.   'JacksonsFiles0803202063717.zip' swiftZips 0

'' swiftZips y
:
```

```
NB. zips in source archive directory !(*)=. dir
zips=. winpathsep&.> 1 dir SwiftZipArchiveDir,'*.zip'

NB. zips after and including (x)
if. #x do.
  if. (<x) e. fnms=. justfileext&.> zips do.
    zips=. (;fnms i.<x) }. zips
  end.
end.

NB. extract sort embedded timestamps
tlen=. #'mmddyyyyhhnnss'
tsms=. >(tlen&{.)@:('JacksonsFiles'&afterstr)&.> zips

NB. yyyy    mm  dd  hr  nn    ss
(\:(4 5 6 7 0 1 2 3 8 9 10 11 12 13) {"1 tsms){zips
)

swiftheader=: 3 : 0

NB.*swiftheader v-- TAB delimited header names from SwiftIQ csv.
NB.
NB. monad:  blcl =. swiftheader clFileCsv
NB.
NB.   path=.'\\jfsdev04\Shares\SwiftIQPreprocess\SwiftZipCsv\'
NB.   csv=. 0 dir path,'Organ*.csv'
NB.   swiftheader path,,csv
NB.
```

```
NB. dyad:  blcl =. clDel swiftheader clFileCsV
NB.
NB.   NB. headers for all directory csvs
NB.   csvs=. 0{1 dir path,'*.csv'
NB.   (justfileext@winpathsep&.> csvs),.CRLF&swiftheader&.> csvs

CRLF swiftheader y
:
'not enough header bytes' assert x +./@E. s=.1!:11 y;0,HEADERLEN <. fsize y

NB. parse after removing chars that wreak headers
h=. <;._1 ',' , (x beforestr s) -. a. -. SWIFTALPHA

NB. make any lower snake case names unique with suffixes
'_' sfxnonunique tolower&.> ' _'&charsub&.> reb&.> h
)

swiftlandcols=: 3 : 0

NB.*swiftlandcols v-- column names in SwiftIQ landing tables.
NB.
NB. monad:  bt =. swiftlandcols uuIgnore

NB. require 'dd' !(*)=. ddcon ddcol dddis
'cannot connect swift_staging' assert 0<ch=. ddcon ODBCSWIFT

tcv=. {{ x ; < ,}.t {"1~ (0{t) i. ;:'COLUMN_NAME' [ t=. x ddcol y  }}
NB. return ordered names
```

```
r=.     'land_Calendar' tcv ch
r=. r ,:'land_FuelProduct' tcv ch
r=. r , 'land_ItemSales' tcv ch
r=. r , 'land_OrganizationView' tcv ch
r=. r , 'land_ProductView' tcv ch
r [ dddis ch
)

tablenamesfrcsvs=: 3 : 0

NB.*tablenamesfrcsvs v-- standard SwiftIQ table name suffixes from csvs.
NB.
NB. monad:  blcl =. tablenamesfrcsvs clPath
NB.
NB.   path=.'\\jfsdev04\Shares\SwiftIQPreprocess\SwiftZipCsv\'
NB.   tablenamesfrcsvs path

NB. j profile !(*)=. dir
('-'&beforestr@(justfileext@winpathsep)&.> y) -.&.> <'_'
)

NB. appends trailing line feed character if necessary
tlf=: ] , ((10{a.)"_ = {:) }. (10{a.)"_

NB. converts character strings to CRLF delimiter
toCRLF=: 2&}.@:;@:((13{a.)&,&.>@<;.1@((10{a.)&,)@toJ)
```

```
NB. converts character strings to host delimiter
toHOST=: toCRLF

NB. converts character strings to J delimiter LF
toJ=: ((10{a.) I.@(e.&(13{a.))@]}  ])@:(#~ -.@((13 10{a.)&E.@,))

todayno=: 3 : 0

NB.*todayno v-- convert dates to day numbers, converse  (todate).
NB.
NB. WARNING: valid only for  Gregorian dates after  and including
NB. 1800 1 1.
NB.
NB. monad:  todayno ilYYYYMMDD
NB.
NB.   dates=. 19530702 19520820 20000101 20000229
NB.   todayno 0 100 100 #: dates
NB.
NB. dyad:  pa todayno itYYYYMMDD
NB.
NB.   1 todayno dates

0 todayno y
:
a=. y
if. x do. a=. 0 100 100 #: a end.
a=. ((*/r=. }: $a) , {:$a) $,a
```

```
'y m d'=. <"_1 |: a
y=. 0 100 #: y - m <: 2
n=. +/ |: <. 36524.25 365.25 *"1 y
n=. n + <. 0.41 + 0 30.6 #. (12 | m-3),"0 d
0 >. r $ n - 657378
)

tolower=: 3 : 0

NB.*tolower v-- convert to lower case.
NB.
NB. monad: cl =. tolower cl

x=. I. 26 > n=. ((65+i.26){a.) i. t=. ,y
($y) $ ((x{n) { (97+i.26){a.) x}t
)

topswiftTrap=: 3 : 0

NB.*topswiftTrap v-- top level SwiftIQ error trapping.
NB.
NB. monad:  iaRc =. topswiftTrap uuIgnore
NB. dyad:  iaRc =. uuIgnore topswiftTrap uuIgnore

(swiftTrap :: swiftFail) 0
:
(swiftTrap :: swiftFail) 0
)
```

```
unzipswift=: 3 : 0

NB.*unzipswift v-- unzips zip files in SwiftIQ working directory.
NB.
NB. monad:  clMsg =. unzipswift uuIgnore

cmd=. 'pushd ',enquote SwiftZipCsvDir
cmd=. cmd,LF,JunzipCmd,' *.zip'
(toHOST cmd) write bat=. SwiftCodeDir,'unzipswift.bat'

NB. require 'task' !(*)=. shell
shell enquote bat
)

NB. day of week - see long document:  weekday 1953 7 2
weekday=: 7 | 3 + todayno

NB. standardizes path delimiter to windows back \ slash
winpathsep=: '\'&(('/' I.@:= ])} )

NB. writes a list of bytes to file
write=: 1!:2 ]`<@.(32&>@(3!:0))

writeswiftHeaders=: 3 : 0

NB.*writeswiftHeaders v-- writes first header row for all SwiftIQ landing tables.
NB.
```

```
NB. monad:  blclFiles =. writeswiftHeaders btLandingCols
NB.
NB.   writeswiftHeaders swiftlandcols 0

files=. (<SwiftTsvDir) ,&.> (0{"1 y) ,&.> <'.txt'
files [ (toHOST@fmttd&.> 1 {"1 y) write&.> files
)

NB.POST_swiftprep post processor.

smoutput IFACE=:  (0 : 0)
NB. (swiftprep) interface word(s): 20210805j85156
NB. ------------------------------
NB. dailyswift           NB. daily SwiftIQ zip loading and maintenance tasks
NB. fullheaderscan       NB. scans all the SwiftIQ zips and builds complete table headers
NB. loadswiftcsvs        NB. copies zip to SwiftIQ working directory and extracts csvs
NB. loadswifttabs        NB. convert SwiftIQ csvs to TAB delimited table files
NB. procqueuezips        NB. processes queued zips from file
NB. procswiftzip         NB. process a SwiftIQ zip
NB. queueswiftSizes      NB. file sizes of SwiftIQ queues: queueswiftSizes SwiftAuxDir,'swift2.txt'
NB. queueswiftZips       NB. write (y) line file of queued zips
NB. runswiftssis         NB. runs SwiftIQ SSIS batch scripts
NB. showswiftUnload      NB. list never loaded zips in load order
NB. showswiftlog         NB. show SwiftIQ log entries
NB. swiftLoaded          NB. loaded swift zips with return code: swiftLoaded '<1000'
NB. swiftMissingColumns  NB. list missing SwiftIQ landing table columns
NB. swiftZips            NB. sorted SwiftIQ zip files
)
```

```
cocurrent 'base'
coinsert  'swiftprep'
```

# Index