ExportAllUserTablesStatus ; Entry point for export status report NEW originalNS SET originalNS=\$NAMESPACE NEW \$ETRAP SET \$ETRAP="GOTO Cleanup"

```
;— 1) Configuration
NEW exportDir,summaryFile,fileObj,sc
SET exportDir="C:\Path\To\ExportCSVs"
SET summaryFile=exportDir_"\"_"ExportStatus.csv"
WRITE !,"*** [1] ExportAllUserTablesStatus starting",!
WRITE "[1] exportDir: ",exportDir,!
WRITE "[1] summaryFile: ",summaryFile,!

;— 2) Ensure output directory exists
WRITE "[2] Ensuring export directory exists...",!
DO ##class(%Library.File).CreateDirectoryChain(exportDir)
WRITE "[2] Directory ready",!

;— 3) Open summary CSV for write
WRITE "[3] Opening summary CSV for write...",!
SET fileObj=##class(%FileCharacterStream).%New()
SET sc=fileObj.Open(summaryFile,"WNS",5)
IF $SYSTEM.Status.IsError(sc)  DO  GOTO Cleanup
. WRITE "[3][ERROR] Cannot open summary CSV → ",summaryFile,!
. DO $SYSTEM.Status.DisplayError(sc)
WRITE "[3] Summary CSV opened",!
DO
fileObj.WriteLine("Namespace,ClassName,DataLocation,GlobalExists,CSVExists,CSVRows,GlobalRows,CSV

;— 4) Enumerate namespaces, skip those containing "MHR"
NEW nsArray
WRITE "[4] Listing namespaces...",!
SET sc=##class(%SYS.Namespace).ListAll(.nsArray)
IF $SYSTEM.Status.IsError(sc)  DO  GOTO Cleanup
. WRITE "[4][ERROR] Cannot list namespaces",!
. DO $SYSTEM.Status.DisplayError(sc)
WRITE "[4] Namespaces retrieved",!

;— Initialize aggregate counters
NEW totalClasses,totalWithData,exportedCount,skippedCount
NEW totalCSVRows,totalGlobalRows,totalCSVSize,totalGlobalSize
SET (totalClasses,totalWithData,exportedCount,skippedCount)=0
SET (totalCSVRows,totalGlobalRows,totalCSVSize,totalGlobalSize)=0

;— 5) Loop each namespace
NEW idx,ns
SET idx=0
FOR  SET idx=$ORDER(nsArray(idx)) QUIT:'idx  DO
```

```
. SET ns=nsArray(idx)
. IF ns["MHR" QUIT
. WRITE !!,"[5] Processing namespace: ",ns,!
. ZN ns
. WRITE "[5] Current namespace: ",$NAMESPACE,!

. ;— 6) Query all non-system classes
. NEW stmt,rs
. SET stmt=##class(%SQL.Statement).%New()
. SET sc=stmt.%Prepare("SELECT Name,DataLocation FROM
%Dictionary.ClassDefinition WHERE System=0")
. IF $SYSTEM.Status.IsError(sc)  DO  QUIT
. . WRITE "[6][ERROR] SQL Prepare failed in ",ns,": "
. . DO $SYSTEM.Status.DisplayError(sc)
. SET rs=stmt.%Execute()
. IF $SYSTEM.Status.IsError(rs)  DO  QUIT
. . WRITE "[6][ERROR] SQL Execute failed in ",ns,": "
. . DO $SYSTEM.Status.DisplayError(rs)
. WRITE "[6] Retrieved classes in ",ns,!

. ;— 7) Loop classes
. FOR  QUIT:'(rs.%Next())  DO
. . NEW className,dataLoc
. . SET className=rs.%Get("Name"),dataLoc=rs.%Get("DataLocation")
. . SET totalClasses=totalClasses+1
. . WRITE "[7] Class: ",className," → DataLocation: ",dataLoc,!

. . ;— 7a) Check global existence & count nodes
. . NEW globalExists,globalRows,glSizeBytes,sub
. . SET (globalExists,globalRows,glSizeBytes)=0
. . IF dataLoc'""  DO
. . . SET sub=""
. . . FOR  SET sub=$ORDER(@(dataLoc_"("_"sub"_")"))  QUIT:sub=""  DO
. . . . SET globalRows=globalRows+1
. . . SET glSizeBytes=##class(%SYS.GlobalQuery).Size(dataLoc)
. . . IF globalRows>0 SET globalExists=1
. . SET totalWithData=totalWithData+globalExists
. . SET totalGlobalRows=totalGlobalRows+globalRows
. . SET totalGlobalSize=totalGlobalSize+glSizeBytes

. . ;— 7b) Check CSV file existence & stats
. . NEW csvFile,csvExists,csvRows,csvSizeBytes,stream,line
. . SET csvFile=exportDir_"\"_ns_"_"_className_".csv"
. . SET (csvExists,csvRows,csvSizeBytes)=0
. . IF ##class(%Library.File).Exists(csvFile)  DO
. . . SET csvExists=1
. . . SET stream=##class(%FileCharacterStream).%New()
. . . SET sc=stream.Open(csvFile,"RS",5)
. . . IF '$SYSTEM.Status.IsError(sc)  DO
. . . . FOR  QUIT:stream.AtEnd  DO  . . . . stream.ReadLine(.line) SET
csvRows=csvRows+1
. . . . SET csvSizeBytes=stream.Size
. . . . DO stream.Close()
. . SET totalCSVRows=totalCSVRows+csvRows
. . SET totalCSVSize=totalCSVSize+csvSizeBytes

. . ;— Compute percentages and status
. . NEW pctRows,pctSize,remainRows,remainSize,status
. . SET pctRows = $SELECT(globalRows>0:(csvRows*100)\globalRows,1:0)
. . SET pctSize = $SELECT(glSizeBytes>0:(csvSizeBytes*100)\glSizeBytes,1:0)
. . SET remainRows=globalRows-csvRows
. . SET remainSize=glSizeBytes-csvSizeBytes
. . SET status=$SELECT('globalExists:"NoData",'csvExists:"NoCSV",
(pctRows=100&&pctSize=100):"Complete",csvExists:"Partial",1:"Pending")
. . IF csvExists SET exportedCount=exportedCount+1
. . ELSE  SET skippedCount=skippedCount+1

. . ;— Write summary record
. . DO
fileObj.WriteLine(ns_","_className_","_dataLoc_","_globalExists_","_csvExists_","_csvRows_","_glo

;— End namespace loop
ZN originalNS
```

Cleanup IF \$DATA(fileObj) DO fileObj.Close() ZN originalNS WRITE !,"*** ExportAllUserTablesStatus complete",! QUIT