

|  |  |
| --- | --- |
|  |  |

Table Tool Examples

CA Services

© 2017 CA

|  |  |  |  |
| --- | --- | --- | --- |
| Document Title | | Table Tool Examples | |
| Customer | |  | |
| Last Saved Date | |  | |
| Version | | 1.1 | |

Table Tool Examples

Contents

[Example #1. Build DB2 Binds for an Element within a processor. 3](#_Toc496550280)

[Example #2. Identify and DELETE old elements in a development environment. 6](#_Toc496550281)

[Example #3a. Build a report of Package information using CSV 9](#_Toc496550282)

[Example #3b. Build a report of Package information – without CSV 12](#_Toc496550283)

[Example #4. Calculate and report information about datasets listed in a saved 3.4 dataset list. 16](#_Toc496550284)

[Example #5. Build Add statements for members of a PDS, where the members may already exist in Endevor with various System, Subsystem and Type names. 19](#_Toc496550285)

[Example #6. Resolve Endevor “Synchronization” errors for one or more elements by executing PDM and replacing the element. 23](#_Toc496550286)

[Example #7. Convert PDS datasets into PDS/E datasets. 33](#_Toc496550287)

[Other example uses of Table Tool: 35](#_Toc496550288)

# Example #1. Build DB2 Binds for an Element within a processor.

Featuring:

1. Table Searching
2. Mixing Endevor variables with Table tool variables
3. Expanding the Model from both Table and Options variable values

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//\*\* GET PROGRAM SPECIFIC OPTIONS ELEMENT (IF IT EXISTS) \*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*This step fetches the program “Options” if one exists*

//GET#OPTN EXEC PGM=IEBUPDTE

//SYSIN DD \*

./ REPRO NEW=PS,NAME=&C1ELEMENT

//SYSUT1 DD DISP=SHR,DSN=WALJO11.ENDEVOR.OPTIONS.&C1STGID,

// MONITOR=COMPONENTS,ALLOC=LMAP

//SYSPRINT DD SYSOUT=\*

//SYSUT2 DD DSN=&&OPTIONS,DISP=(,PASS),

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

*Table Tool is used to Build DB2 Binds for the element being processed. A table of DB2 information by Stage is searched.*

//\*\* Build DB2 Bind Statement(s) \*\*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//BLDBINDS EXEC PGM=IRXJCL,PARM='ENBPIU00 M &C1STAGE'

//TABLE DD \*

\* STG-- DB2Subsystem

*Endevor variables are replaced before Table Tool runs*

TEST DB2T

QA DB2Q

PROD DB2P

//MODEL DD \*

\*\*\* Bind for the &C1STAGE Stage \*/

*Table Tool variables are replaced by values from the Table or by values assigned in OPTIONS*

DSN SYSTEM(&DB2Subsystem)

BIND PACKAGE(&DB2Package) +

OWNER(&DB2Owner) +

RETAIN VALIDATE(BIND) ISOLATION(CS) +

ACTION(REPLACE) EXPLAIN(YES) +

*Default values may be overridden at the element level*

*Example OPTIONS statements:*

*IGYCRCTL='NOSEQ,OBJECT,APOST,DATA(24)'*

*ASMA90='LIB,NOSEQ,OBJECT'*

*IEWL='LIST,MAP,XREF,AMODE(31),RMODE(24)'*

*TESTDB2Package = 'PKGT'*

*TESTDB2Owner = 'OWNT'*

*QADB2Package = 'PKGQ'*

*QADB2Owner = 'OWNQ'*

*PRODDB2Package = 'PKGP'*

*PRODDB2Owner = 'OWNP'*

MEMBER (&C1ELEMENT)

//\* / Site Defaults \

//OPTIONS DD DISP=SHR,DSN=WALJO11.ENDEVOR.OPTIONS.DFLT($$2$DFLT)

//\* / Program Overrides \

// DD DSN=&&OPTIONS,DISP=(OLD,DELETE)

// DD \*

DB2Package = &C1STAGE.DB2Package

DB2Owner = &C1STAGE.DB2Owner

//SYSTSPRT DD SYSOUT=\*

//SYSEXEC DD DISP=SHR,DSN=&CSIQCLS0

//TBLOUT DD DSN=&&DB2BINDS,DISP=(,PASS),

*The output feeds a Bind Step*

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FB,LRECL=80,BLKSIZE=27920)

//\*--------------------------------------------------------------------

//\*--- Execute Binds under Endevor Alternate ID -----------------------

//\*--------------------------------------------------------------------

//DB2BIND EXEC PGM=IKJEFT1B,MAXRC=4,

// COND=((0,NE,BLDBINDS),(4,LT))

//STEPLIB DD DSN=&SDSNEXIT,DISP=SHR

// DD DSN=&SDSNLOAD,DISP=SHR

//DBRMLIB DD DISP=SHR,DSN=&HLQ.&C1ST..&SYSCODE...DBRM,

// MONITOR=COMPONENTS,ALLOC=LMAP

//SYSTSIN DD \* Swap to Endevor Alternate Id

EXEC 'NDVW.PR.APROD.ENDVR18.CSIQCLS0(SWAP2ALT)'

// DD DSN=&&DB2BINDS,DISP=(OLD,DELETE)

// DD \* Swap back to User's Id

EXEC 'NDVW.PR.APROD.ENDVR18.CSIQCLS0(SWAP2USR)'

//SYSTSPRT DD DISP=(OLD,PASS),DSN=&&DB2BNOL

**Sample output:**

\*\*\* Bind for the TEST Stage \*/

DSN SYSTEM(DB2T)

BIND PACKAGE(PKGT) +

OWNER(OWNT) +

RETAIN VALIDATE(BIND) ISOLATION(CS) +

ACTION(REPLACE) EXPLAIN(YES) +

MEMBER (TESTPGMX)

# Example #2. Identify and DELETE old elements in a development environment.

Featuring:

1. The user designates what is considered “old”
2. Endevor CSV utility for element information
3. REXX calculations within OPTIONS

//\*-------------------------------------------------------------------

// SET SYSEXEC=CAPRD.NDVR.PROD.CATSNDVR.CEXEC

//\*--------------------------------------------------------------

//\*- To Report and delete very old elements from Test -----------

//\*--------------------------------------------------------------------\*

//\* STEP 1 -- Execute CSV Utility to locate inventory

//\*--------------------------------------------------------------------\*

*This step executes the Endevor CSV utility and outputs element information.*

//STEP1 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTU

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTH

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQLOAD

//BSTIPT01 DD \*

LIST ELEMENT '\*'

FROM ENVIRONMENT SMPLTEST SYSTEM '\*' SUBSYSTEM '\*'

TYPE '\*'

*The utility is documented in* ***Chapter 13: Using the Comma Separated Value (CSV) Utility*** *of the* ***Endevor Utilities Guide.***

*This example shows one of the many supported requests.*

STAGE NUMBER '\*'

DATA BASIC

TO DDNAME 'CSVOUTPT'

OPTIONS NOSEARCH RETURN FIRST .

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

*The output is produced in the Comma Separated Value format*

//CSVOUTPT DD DSN=&&CSVFILE,

// DCB=(RECFM=FB,LRECL=1800,BLKSIZE=9000,DSORG=PS),

// DISP=(MOD,PASS),

// SPACE=(CYL,(5,5),RLSE)

//\*--------------------------------------------------------------------

//\* SHOWME -- Show the API call results

//\*--------------------------------------------------------------------

*This step shows the CSV output.*

//SHOWME EXEC PGM=IEBGENER,REGION=1024K,COND=(4,EQ,STEP1)

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&CSVFILE,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------

//\*

*The Table Tool (ENBPIU00) step reads the CSV file and writes out DELETE SCL statements.*

//TAILOR EXEC PGM=IRXJCL,PARM='ENBPIU00 A'

//TABLE DD DSN=&&CSVFILE,DISP=(OLD,DELETE)

//OPTIONS DD \*

*OPTIONS may execute any single-line REXX statement. This example uses built-in REXX routines to calculate the age of the element from its UPDT\_DATE value within the CSV data.*

*The value given to* ***DaysAgo*** *designates the definition of “Old”.*

***These REXX statements determine whether or not to build DELETE SCL for the element.***

**DaysAgo = 240 /\* Number of days for cutoff \*/**

**BaseDate = DATE('B') /\* Today in Base format \*/**

**UpdateDate = Substr(UPDT\_DATE,1,4) || Substr(UPDT\_DATE,6,2)**

**UpdateDate = UpdateDate || Substr(UPDT\_DATE,9,2)**

**If DATATYPE(UpdateDate) /= 'NUM' then $SkipRow = 'Y'**

**BaseOld = DATE(B,UpdateDate,S) /\* Convert Upd date to Base fmt \*/**

**ElementAge = BaseDate - BaseOld /\* Determine how many days ago \*/**

**If TYPE\_NAME /= 'JAVA' & ElementAge < DaysAgo then $SkipRow = 'Y'**

//SYSEXEC DD DSN=&SYSEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//MODEL DD \*

\*\* **&FULL\_ELM\_NAME** &SYS\_NAME &SBS\_NAME &TYPE\_NAME &SIGNOUT\_ID

*Notice the use of the variable* ***FULL\_ELM\_NAME*** *versus ELM\_NAME.*

\* &UPDT\_DATE &UPDT\_TIME (&ElementAge Days ago)

DELETE ELEMENT **&FULL\_ELM\_NAME**

FROM ENVIRONMENT &ENV\_NAME

SYSTEM &SYS\_NAME SUBSYSTEM &SBS\_NAME

TYPE &TYPE\_NAME STAGE &STG\_ID .

//TBLOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSIN DD DUMMY

//TBLOUT DD SYSOUT=\*

//\*-------------------------------------------------------------------

**Sample output:**

\*\* ElapsedDaysCalcs CONCURNT ACCTPAY JAVA OCLEO01

\* 2014/06/19 07:49:44:26 (314 Days ago)

DELETE ELEMENT ElapsedDaysCalcs

FROM ENVIRONMENT SMPLTEST

SYSTEM CONCURNT SUBSYSTEM ACCTPAY

TYPE JAVA STAGE T .

\*\* calculateElapsedDays CONCURNT ACCTPAY JAVA OCLEO01

\* 2015/04/28 19:01:07:26 (1 Days ago)

DELETE ELEMENT calculateElapsedDays

FROM ENVIRONMENT SMPLTEST

SYSTEM CONCURNT SUBSYSTEM ACCTPAY

TYPE JAVA STAGE T .

\*\* ElapsedDaysPanel CONCURNT ACCTPAY JAVA WALJO11

\* 2015/04/28 19:27:37:37 (1 Days ago)

DELETE ELEMENT ElapsedDaysPanel

FROM ENVIRONMENT SMPLTEST

SYSTEM CONCURNT SUBSYSTEM ACCTPAY

TYPE JAVA STAGE T .

\*\* CONC0003 CONCURNT CONTROLL ALIAS WALJO11

\* 2013/12/09 19:54:45:29 (506 Days ago)

DELETE ELEMENT CONC0003

FROM ENVIRONMENT SMPLTEST

SYSTEM CONCURNT SUBSYSTEM CONTROLL

TYPE ALIAS STAGE T .

# Example #3a. Build a report of Package information using CSV

Featuring:

1. Endevor CSV utility for package information
2. Using a pass within a PARMLIST to initialize variables
3. A MODEL and REXX statements that format a report
4. A call to the internal ENBPIU00 routine named **BuildFromMODEL**

// SET SYSEXEC=CAPRD.NDVR.PROD.CATSNDVR.CEXEC

//\*--------------------------------------------------------------

//\*- To Report Packages created over nnn days ago -----------

//\*--------------------------------------------------------------------\*

//\* STEP 1 -- Execute CSV Utility to gather Package information

//\*--------------------------------------------------------------------\*

*This step executes the Endevor CSV utility and outputs Package information.*

//STEP1 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTU

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTH

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQLOAD

//BSTIPT01 DD \*

*The utility is documented in* ***Chapter 13: Using the Comma Separated Value (CSV) Utility*** *of the* ***Endevor Utilities Guide.***

*This example shows one of many supported requests.*

LIST PACKAGE ID '\*'

WHERE DATE TYPE CR IS OLDER THAN 20 DAYS

TO DDNAME 'CSVOUTPT' .

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

*The output is produced in the Comma Separated Value format*

//CSVOUTPT DD DSN=&&CSVFILE,

// DCB=(RECFM=FB,LRECL=1800,BLKSIZE=9000,DSORG=PS),

// DISP=(MOD,PASS),

// SPACE=(CYL,(5,5),RLSE)

//\* SHOWME -- Show the API call results ---------------------------------------------------

*To print the intermediate results*

//SHOWME EXEC PGM=IEBGENER,REGION=1024K,COND=(4,EQ,STEP1)

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&CSVFILE,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------

//REPORT EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

*The Table Tool (ENBPIU00) step reads the CSV file and writes a report.*

//TABLE DD DSN=&&CSVFILE,DISP=(OLD,DELETE)

//PARMLIST DD \*

MODEL TBLOUT OPTION0 0

*The first pass of the table reads 0 records, but executes the REXX statements in OPTIONS0 once.*

MODEL TBLOUT OPTIONS A

//**HEADING** DD \*

\* Package--------- Status----- CreateDate UpdateDate CreatorId- PackageAge

//\*-+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8

//**MODEL** DD \*

*The format of the output is determined by two MODELs. The first one named* ***HEADING*** *builds the report heading.*

*The second one,* ***MODEL****, formats the detail lines with help from the OPTIONS.*

&DetailLine

//**OPTION0** DD \*

LinesPerPage = 15

LineCount = LinesPerPage + 1 /\*Cause first page heading \*/

DaysAgo = 60 /\* Number of days for cutoff \*/

//**OPTIONS** DD \*

\* Bypass processing for Table header

***OPTIONS*** *statements may execute any single-line REXX statement. The statements in this example use built-in REXX routines to calculate the age of a Package.*

***Table Tool executes the OPTIONS once before processing the first row of the table.***

**If $row# < 1 then $SkipRow = 'Y'**

\* Calculate age of Package creation

BaseDate = DATE('B') /\* Today in Base format \*/

UpdateDate = Substr(UPDT\_DATE,1,4) || Substr(UPDT\_DATE,6,2)

UpdateDate = UpdateDate || Substr(UPDT\_DATE,9,2)

If DATATYPE(UpdateDate) /= 'NUM' then $SkipRow = 'Y'

BaseOld = DATE(B,UpdateDate,S) /\* Convert Upd\_date to Base fmt \*/

PackageAge = BaseDate - BaseOld /\* Determine how many days ago \*/

If PackageAge < DaysAgo then $SkipRow = 'Y' /\* Skip if recent \*/

\* Build report detail line ....

**DetailLine = Copies(' ',120);**

**DetailLine = Overlay(PKG\_ID,DetailLine,03)**

***Additional REXX statements format the detail line of the report****.*

**DetailLine = Overlay(STATUS,DetailLine,20)**

**DetailLine = Overlay(CREATE\_DATE,DetailLine,32)**

**DetailLine = Overlay(UPDT\_DATE,DetailLine,43)**

**DetailLine = Overlay(CREATE\_USRID,DetailLine,54)**

**DetailLine = Overlay(PackageAge,DetailLine,65)**

\* Determine whether it is time for page heading

***This statement causes the report heading to print.***

***BuildFromModel is an internal routine of ENBPIU00 that can be called within OPTIONS.***

***The parameter in Parentheses may be any DDName or MODEL you want written.***

LineCount = LineCount + 1

**If LineCount > LinesPerPage then x = BuildFromMODEL(HEADING)**

If LineCount > LinesPerPage then LineCount = 1

//SYSEXEC DD DSN=&SYSEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSIN DD DUMMY

//TBLOUT DD SYSOUT=\*

# Example #3b. Build a report of Package information – without CSV

Featuring:

1. Endevor CSV utility for package information – with the NOCSV option
2. Using a pass within a PARMLIST to initialize variables
3. A MODEL and REXX statements that format a report
4. A call to the internal ENBPIU00 routine named **BuildFromMODEL**

//\*--------------------------------------------------------------------\*

//\* STEP 1 -- Execute CSV Utility to gather Package information

//\*--------------------------------------------------------------------\*

*This step executes the Endevor CSV utility and outputs Package information in an API format.*

//STEP1 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTU

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTH

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQLOAD

//BSTIPT01 DD \*

*The utility is documented in* ***Chapter 13: Using the Comma Separated Value (CSV) Utility*** *of the* ***Endevor Utilities Guide.***

*This example shows one of many supported requests.*

*Output is generated in a fixed format – not a CSV format.*

LIST PACKAGE ID '\*'

WHERE DATE TYPE CR IS OLDER THAN 20 DAYS

TO DDNAME 'CSVOUTPT'

OPTIONS **NOCSV**.

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

*The output is produced in the fixed format*

//CSVOUTPT DD DSN=&&CSVFILE,

// DCB=(RECFM=FB,LRECL=1800,BLKSIZE=9000,DSORG=PS),

// DISP=(MOD,PASS),

// SPACE=(CYL,(5,5),RLSE)

//\*--------------------------------------------------------------------

//\* SHOWME -- Show the API call results

//\*--------------------------------------------------------------------

*To print the intermediate results*

//SHOWME EXEC PGM=IEBGENER,REGION=1024K,COND=(4,EQ,STEP1)

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&CSVFILE,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------

*The Table Tool (ENBPIU00) step reads the fixed format file and writes a report.*

//REPORT EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

//TABLE DD DSN=&&CSVFILE,DISP=(OLD,DELETE)

//POSITION DD \*

ALPKG\_RS\_PKGID 13 28

ALPKG\_RS\_SITE 29 29

ALPKG\_RS\_COMMENT 30 79

*The positions of fields may be defined within the* ***POSITION*** *data when the TABLE is using a fixed format*

ALPKG\_RS\_PKG\_TYPE 80 89

ALPKG\_RS\_STAT 116 127

ALPKG\_RS\_CRD 155 161

ALPKG\_RS\_CRU 167 174

ALPKG\_RS\_MOD 175 181

//PARMLIST DD \*

*The first pass of the table reads 0 records, but executes the REXX statements in OPTIONS0 once.*

MODEL TBLOUT OPTION0 0

MODEL TBLOUT OPTIONS A

//HEADING DD \*

\* Package--------- Status----- CreateDate UpdateDate CreatorId- PackageAge

//\*-+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8

//MODEL DD \*

*The format of the output is determined by two MODELs. The first one named* ***HEADING*** *builds the report heading.*

*The second one,* ***MODEL****, formats the detail lines with help from the OPTIONS.*

&DetailLine

//OPTION0 DD \*

LinesPerPage = 15

LineCount = LinesPerPage + 1

DaysAgo = 10 /\* Number of days for cutoff \*/

Months='JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV DEC'

//OPTIONS DD \*

***OPTIONS*** *statements may execute any single-line REXX statement. The statements in this example use built-in REXX routines to calculate the age of a Package.*

***Table Tool executes the OPTIONS once before processing the first row of the table.***

\* Bypass processing for Table header

If $row# < 1 then $SkipRow = 'Y'

\* Calculate age of Package creation

BaseDate = DATE('B') /\* Today in Base format \*/

year = Substr(ALPKG\_RS\_MOD,6,2)

If DATATYPE(year) /= 'NUM' then $SkipRow = 'Y'

if year < 50 then UpdateDate='20'year

if year >= 50 then UpdateDate='19'year

Say 'UpdateDate =' UpdateDate ALPKG\_RS\_PKGID

month = Substr(ALPKG\_RS\_MOD,3,3)

month# = WordPos(month,Months)

month# = Right(month#,2,'0')

Say 'UpdateDate =' UpdateDate ALPKG\_RS\_PKGID

UpdateDate = UpdateDate || month#

UpdateDate = UpdateDate || Substr(ALPKG\_RS\_MOD,1,2)

Say 'UpdateDate =' UpdateDate ALPKG\_RS\_PKGID

BaseOld = DATE(B,UpdateDate,S) /\* Convert Upd date to Base fmt \*/

PackageAge = BaseDate - BaseOld /\* Determine how many days ago \*/

If PackageAge < DaysAgo then $SkipRow = 'Y' /\* Skip if recent \*/

***Additional REXX statements format the detail line of the report****.*

\* Build report detail line ....

DetailLine = Copies(' ',120);

DetailLine = Overlay(ALPKG\_RS\_PKG\_TYPE,DetailLine,03)

DetailLine = Overlay(ALPKG\_RS\_STAT,DetailLine,20)

DetailLine = Overlay(ALPKG\_RS\_CRD,DetailLine,32)

DetailLine = Overlay(ALPKG\_RS\_MOD,DetailLine,43)

DetailLine = Overlay(ALPKG\_RS\_CRU,DetailLine,54)

DetailLine = Overlay(PackageAge,DetailLine,65)

***This statement causes the report heading to print.***

***BuildFromModel is an internal routine of ENBPIU00 that can be called within OPTIONS.***

***The parameter in Parentheses may be any DDName or MODEL you want written.***

\* Determine whether it is time for page heading

LineCount = LineCount + 1

If LineCount > LinesPerPage then x = BuildFromMODEL(HEADING)

If LineCount > LinesPerPage then LineCount = 1

//SYSEXEC DD DSN=&SYSEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSIN DD DUMMY

//TBLOUT DD SYSOUT=\*

//\*-------------------------------------------------------------------

**Sample output:**

\* Package--------- Status----- CreateDate UpdateDate CreatorId- PackageAge

E#OC1M5522904977 COMMITTED 2014/03/27 2014/12/23 WALJO11 91

E#OC1N3755897106 COMMITTED 2014/03/27 2014/12/23 WALJO11 91

E#OD3I0026463733 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3J4915633577 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3J5233662270 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3K1724536427 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3K3545190469 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3N5123624462 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3O0142171323 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3O1021444274 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3O1238739916 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3O1357412794 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD3O5446124312 COMMITTED 2014/04/29 2014/12/23 WALJO11 91

E#OD4O1827140268 COMMITTED 2014/04/30 2014/12/23 WALJO11 91

E#OD4O3115929537 COMMITTED 2014/04/30 2014/12/23 WALJO11 91

\* Package--------- Status----- CreateDate UpdateDate CreatorId- PackageAge

EOINBACKOUTTHISP EXECUTED 2014/07/30 2014/07/30 OCLEO01 237

EOINBACKOUTTHIS2 EXECUTED 2014/07/30 2014/07/30 OCLEO01 237

EOINNEW IN-EDIT 2014/06/27 2014/06/27 OCLEO01 270

EOINTEST IN-EDIT 2014/09/02 2014/09/02 OCLEO01 203

ESYM141123165204 EXECUTED 2014/11/23 2014/11/23 WALJO11 121

ESYM141123172333 EXECUTED 2014/11/23 2014/11/23 WALJO11 121

ESYM141123203527 EXECUTED 2014/11/23 2014/11/23 WALJO11 121

ESYM141123210824 EXECUTED 2014/11/23 2014/11/23 WALJO11 121

ESYM141205171050 EXECUTED 2014/12/05 2014/12/05 WALJO11 109

# Example #4. Calculate and report information about datasets listed in a saved 3.4 dataset list.

*These are some of the fields in the Saved dataset list.*

Featuring:

1. Nothing to do with Endevor
2. TSO’s 3.4 - Data Set List Utility
3. POSITION data for an input table

//\*--------------------------------------------------------------------

//\*- Report the total Track consumption of a list of datasets ---------

//\*- in a list created from a saved TSO 3.4 list. ---------

//\*- Also identify and delete old datasets . ---------

//\*--------------------------------------------------------------------

// SET SYSEXEC=SYS1.EXEC <- prd Version

//\*--------------------------------------------------------------------

*The Table Tool (ENBPIU00) executed as interpreted REXX.*

//TEST001 EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

//SYSEXEC DD DISP=SHR,DSN=&SYSEXEC

*This dataset is created by:*

* *Using TSO option 3.4 to list datasets*
* *Entering “Save” and a name on the command line*
* *TSO creates a dataset named <yourUserid>.<name>.DATASETS.*

//TABLE DD DISP=SHR,DSN=WALJO11.LIST3#4.DATASETS

//PARMLIST DD \*

NOTHING NOTHING **OPTIONS0** 0

**MODEL1** TBLOUT1 **OPTIONS1** A

**MODEL2** TBLOUT2 **OPTIONS2** A

//POSITION DD \*

Dataset 01 44

*Field positions in the SAVEd dataset*

Dsorg 53 54

Tracks 80 88

CreateDate 108 117

AccessDate 130 139

These options are executed only once at the beginning, and set initial values for REXX variables.

//OPTIONS0 DD \*

$StripData = 'N' ; /\* Preserve spaces \*/

BaseDate = DATE('B') /\* Today in Base format \*/

DaysAgo = 240 /\* Number of days for cutoff \*/

Total=0 /\* Initialize variable \*/

//OPTIONS1 DD \*

\* Determine how old is the dataset

Using a date field in the input, calculate the age of the dataset

Parse Var CreateDate yr '/' mo '/' da

date = yr || mo || da

If DATATYPE(date) /= 'NUM' then $SkipRow = 'Y'

BaseOld = DATE(B,date,S) /\* Convert Upd date to Base fmt \*/

ElementAge = BaseDate - BaseOld /\* Determine how many days ago \*/

ElementAge = Right(ElementAge,4,'0') /\* For fixed width \*/

//OPTIONS2 DD \*

*Table Tool executes the OPTIONS once before processing the first row of the table*

\* Calculate a running count of Tracks consumed

If $row#<1 then $SkipRow = 'Y'

If DATATYPE(Tracks) /= 'NUM' Then $SkipRow = 'Y'

Total = Total + Tracks

Using the number of Tracks given for each dataset, keep a running total of Tracks consumed.

**Adjust the return code for this step based on the Track total exceeding certain thresholds.**

Total = Right(Total,6,'0') /\* For fixed width \*/

Tracks = Right(Tracks,6,'0') /\* For fixed width \*/

**If Total > 1000 then $my\_rc = 2**

**If Total > 5000 then $my\_rc = 4**

//MODEL1 DD \*

&Dataset Date=&CreateDate DaysOld=&ElementAge

//MODEL2 DD \*

&Dataset Total=&Total Tracks=&Tracks

//TBLOUT1 DD SYSOUT=\*

//TBLOUT2 DD SYSOUT=\*

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//DISPLAYS DD SYSOUT=\*

//SYSTSIN DD DUMMY

//\*--------------------------------------------------------------------

**Sample output from pass #1:**

WALJO11.#DSNS Date=2013/11/18 DaysOld=0501

WALJO11.#DSNS.TSO06 Date=2013/12/20 DaysOld=0469

WALJO11.ACM#BILD.#001.ACMCOMP Date=2014/12/30 DaysOld=0094

WALJO11.ACM#BILD.#001.ACMCOMP.X Date=2014/12/30 DaysOld=0094

WALJO11.ACMBILD Date=2014/12/30 DaysOld=0094

WALJO11.ABCCO.JCL Date=2013/07/30 DaysOld=0612

WALJO11.ABCCO.NOTES Date=2013/07/25 DaysOld=0617

WALJO11.ABCCO.PACKAGE Date=2013/10/17 DaysOld=0533

WALJO11.ABCCO.SOURCE Date=2013/10/17 DaysOld=0533

WALJO11.ALIAS.UPT Date=2013/12/11 DaysOld=0478

WALJO11.ANALYSIS.CNTL Date=2011/06/29 DaysOld=1374

WALJO11.ANALYSIS.LOAD Date=2012/11/01 DaysOld=0883

WALJO11.ANALYZER.FAILURE Date=2012/08/28 DaysOld=0948

WALJO11.ANALYZER.SUCCESS Date=2012/08/28 DaysOld=0948

WALJO11.BASE1.LOADLIB Date=2012/11/01 DaysOld=0883

WALJO11.BASE2.LOADLIB Date=2013/12/20 DaysOld=0469

WALJO11.BASE3.LOADLIB Date=2013/12/20 DaysOld=0469

**Sample output from pass #2:**

WALJO11.#DSNS.TSO06 Total=000075 Tracks= 75

WALJO11.ACM#BILD.#001.ACMCOMP Total=000090 Tracks= 15

WALJO11.ACM#BILD.#001.ACMCOMP.X Total=000105 Tracks= 15

WALJO11.ACMBILD Total=000107 Tracks= 2

WALJO11.ABCCO.JCL Total=000122 Tracks= 15

WALJO11.ABCCO.NOTES Total=000137 Tracks= 15

WALJO11.ABCCO.PACKAGE Total=000139 Tracks= 2

WALJO11.ABCCO.SOURCE Total=000140 Tracks= 1

WALJO11.ALIAS.UPT Total=000141 Tracks= 1

WALJO11.ANALYSIS.CNTL Total=000146 Tracks= 5

WALJO11.ANALYSIS.LOAD Total=000161 Tracks= 15

WALJO11.ANALYZER.FAILURE Total=000166 Tracks= 5

WALJO11.ANALYZER.SUCCESS Total=000171 Tracks= 5

WALJO11.BASE1.LOADLIB Total=000186 Tracks= 15

WALJO11.BASE2.LOADLIB Total=000201 Tracks= 15

# Example #5. Build Add statements for members of a PDS, where the members may already exist in Endevor with various System, Subsystem and Type names.

Featuring:

1. Using a pass within a PARMLIST to initialize variables
2. Using a REXX Stem array to provide data from a table into OPTIONS for the search of another table
3. POSITION data

//\*--------------------------------------------------------------

//\*- Add members of a PDS into Endevor -----------

//\*--------------------------------------------------------------

//\*-------------------------------------------------------------------

//\* STEP 1 - Get a list of members in the dataset

*This step provides a list of members within a dataset.*

//\*-------------------------------------------------------------------

//STEP01 EXEC PGM=IKJEFT1B

//SYSTSIN DD \*

*An IBM TSO command*

LISTDS 'WALJO11.ENDEVOR.SOURCE' MEMBERS

//SYSPRINT DD SYSOUT=\*

//SYSTSPRT DD DSN=&&MBRLIST,DISP=(,PASS),

Member list from the IBM utility

// SPACE=(CYL,(1,1)),UNIT=SYSDA,

// LRECL=120,RECFM=FB,BLKSIZE=0

//\*

//\*--------------------------------------------------------------------\*

//\* STEP 2 -- Update Rexx Stem array for each member

//\*--------------------------------------------------------------------\*

*The Table Tool (ENBPIU00) executed as interpreted REXX.*

//STEP02 EXEC PGM=IRXJCL,PARM='ENBPIU00 A'

//TABLE DD DSN=&&MBRLIST,DISP=(OLD,DELETE)

//POSITION DD \*

Member 3 10

//MODEL DD \*

*This Step assigns values to Rexx stem variables. The output from this step is used as OPTIONS in a subsequent step*

FoundMember.&Member = 'Y'

//OPTIONS DD \*

If $row# < 8 then CSVoption = ' '

If $row# < 8 then $SkipRow = 'Y' /\* Skip if recent \*/

Member = Strip(Member)

firstchar = Substr(Member,1,1)

if firstchar < 'A' | firstchar > 'Z' then $SkipRow = 'Y'

*This statement eliminates backout member names – just in case the dataset contains them.*

If $row# > 8 then CSVoption = 'NOCSV'

//SYSEXEC DD DSN=&SYSEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//SYSIN DD DUMMY

*The output contains REXX stem assignment statements. For example, if TESTING is a member in the dataset then output will contain this statement:*

*FoundMember.TESTING = ‘Y’*

//SYSPRINT DD SYSOUT=\*

//TBLOUT DD DSN=&&STEMARRY,DISP=(MOD,PASS),

// SPACE=(CYL,(1,1)),UNIT=SYSDA,

// LRECL=80,RECFM=FB,BLKSIZE=0

//\*--------------------------------------------------------------------

//\* SHOWME -- Show the Stem array settings

//\*--------------------------------------------------------------------

*To print the intermediate results*

//SHOWME EXEC PGM=IEBGENER,REGION=1024K

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&STEMARRY,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

//\* STEP 3 -- Execute CSV Utility to locate inventory

//\*--------------------------------------------------------------------\*

*The Endevor CSV utility to build a CSV file of production elements*

//STEP03 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTU

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQAUTH

// DD DISP=SHR,DSN=CAPRD.NDVR.V160PRD.CSIQLOAD

//BSTIPT01 DD \*

LIST ELEMENT '\*' FROM ENVIRONMENT SMPLPROD STAGE P

SYSTEM '\*' SUBSYSTEM '\*' TYPE "\*"

OPTIONS NOSEARCH TO FILE CSVOUTPT .

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

//CSVOUTPT DD DSN=&&CSVFILE,

*This file is a CSV extract of all production elements*

// DCB=(RECFM=FB,LRECL=1800,BLKSIZE=9000,DSORG=PS),

// DISP=(MOD,PASS),

// SPACE=(CYL,(5,5),RLSE)

//\*--------------------------------------------------------------------

//\* SHOWME -- Show the API call results

//\*--------------------------------------------------------------------

*To print the intermediate results*

//SHOWME EXEC PGM=IEBGENER,REGION=1024K

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&CSVFILE,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------

//\*

*The Table Tool (ENBPIU00) executed as interpreted REXX.*

//STEP04 EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

//TABLE DD DSN=&&CSVFILE,DISP=(OLD,DELETE)

*The CSV extract of all production elements*

//PARMLIST DD \*

NOTHING NOTHING STEMARRY 0

MODEL TBLOUT OPTIONS A

*The Stem array variable assignments*

//STEMARRY DD \* < Build Stem Array FoundMember.

FoundMember. = 'N'

// DD DSN=&&STEMARRY,DISP=(OLD,DELETE)

//NOTHING DD DUMMY

//MODEL DD \*

\*\* &ELM\_NAME &SYS\_NAME &SBS\_NAME &TYPE\_NAME &SIGNOUT\_ID

*An Endevor ADD statement is written for each member found in the dataset and also found as a production element.*

ADD ELEMENT &ELM\_NAME

TO ENVIRONMENT SMPLTEST

SYSTEM &SYS\_NAME SUBSYSTEM &SBS\_NAME

TYPE &TYPE\_NAME.

//OPTIONS DD \*

ELM\_NAME = Strip(ELM\_NAME)

*Stem array variable assignments make the selections for output. An English translation of this REXX statement is*

***“if the element listed in the CSV data was not found in the dataset then bypass it”***

**If FoundMember.ELM\_NAME /= 'Y' then $SkipRow = 'Y'**

//SYSEXEC DD DSN=&SYSEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//TBLOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSIN DD DUMMY

//TBLOUT DD SYSOUT=\*

//\*-------------------------------------------------------------------

**Sample output:**

\*\* C1UEXT02 CATSNDVR CCIDRUSE COBOL WALJO11

ADD ELEMENT C1UEXT02

TO ENVIRONMENT SMPLTEST

SYSTEM CATSNDVR SUBSYSTEM CCIDRUSE

TYPE COBOL.

\*\* C1UEXT03 CATSNDVR ADMIN COBOL

ADD ELEMENT C1UEXT03

TO ENVIRONMENT SMPLTEST

SYSTEM CATSNDVR SUBSYSTEM ADMIN

TYPE COBOL.

\*\* ISITHERE CATSNDVR ADMIN REXX

ADD ELEMENT ISITHERE

TO ENVIRONMENT SMPLTEST

SYSTEM CATSNDVR SUBSYSTEM ADMIN

TYPE REXX.

\*\* ENHAESCL CATSNDVR ENDEVOR ASM

ADD ELEMENT ENHAESCL

TO ENVIRONMENT SMPLTEST

SYSTEM CATSNDVR SUBSYSTEM ENDEVOR

TYPE ASM.

# Example #6. Resolve Endevor “Synchronization” errors for one or more elements by executing PDM and replacing the element.

Featuring:

1. Batch Endevor and CSV utility executions
2. External MODEL
3. Table Tool submission of multiple jobs
4. POSITION data for C1MSGS2 output
5. Use of $delimiter
6. Execution(s) of PDM

//\*-------------------------------------------------------------------\*

//\* Execute Endevor VALIDATE actions for selected inventory

//\* Execute Endevor CSV for elements that receive RC=0008 from Validate

//\* Capture Base/level information, CCID and COMMENT fields from CSV

//\* Submit batch PDM job for each selected element

//\* If SYNC errors are found, this job gets RC=12

//\*-------------------------------------------------------------------\*

//\*---- Select inventory location to fix: -\*

// SET ENVIRON=ADMIND

*Control values are established at the top of the JCL.*

*The user needs only to set the values for each run.*

// SET SYSTEM=ADMIN (Wildcarding is OK)

// SET SUBSYS=\* (Wildcarding is OK)

// SET ELEMENT=\* (Wildcarding is OK)

// SET TYPE=REX (Wildcarding is OK)

// SET STGID=\* (Wildcarding is OK)

//\*---- Designate the production Environment and Stage (for PDM inp) -\*

// SET PENVIRON=ADMIN

// SET PSTGID=Z

//\*---- Select File names for PDM WIP, saved and Merged source -\*

// SET SAVE#LIB='SW.ENDV.SYA.ENDEVOR.SOURCE' <- Saved source here

// SET MRGE#LIB='SW.ENDV.SYA.ENDEVOR.MERGED' <- PDM merges here

// SET PDM#WIP='SW.ENDV.SYA.ENDEVOR.WIP' <- PDM WIP file

//\*--------------------------------------------------------------------\*

Table tool executed as compiled REXX

//RETROFT1 EXEC PGM=ENBPIU00,

// PARM='M &ENVIRON &SYSTEM &SUBSYS &ELEMENT &TYPE &STGID'

*JCL parameter values passed to Table Tool are written to the output*

//TABLE DD \*

\* Environ System Subsys Element Type Stgid MODEL--- TBLOUT--

\* \* \* \* \* \* VALIDMDL VALIDATE

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

//VALIDMDL DD \*

*This Model specifies that the output is to be formatted into an Endevor VALIDATE statement.*

VALIDATE ELEMENT "&Element"

FROM ENVIRONMENT "&Environ"

SYSTEM "ADMIN" SUBSYSTEM "&Subsys"

TYPE "&Type" STAGE "&Stgid"

.

//OPTIONS DD DUMMY

//SYSTSPRT DD SYSOUT=\*

*A Validate Statement is located here*

//VALIDATE DD DSN=&&VALIDATE,DISP=(NEW,PASS),

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=08000)

//\*--------------------------------------------------------------------\*

Endevor batch

//RETROFT2 EXEC PGM=NDVRC1,

// DYNAMNBR=1500,

// REGION=4096K,

// PARM='C1BM3000'

*Execute the Validate Statement*

//BSTIPT01 DD DSN=&&VALIDATE,DISP=(OLD,DELETE)

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//SYSTSPRT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//C1MSGS1 DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

*The summary from the VALIDATE is captured*

//C1MSGS2 DD DSN=&&C1MSGS2,DISP=(NEW,PASS),

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FBA,LRECL=133,BLKSIZE=26600)

//\*--------------------------------------------------------------------\*

Table tool executed as compiled REXX

//RETROFT3 EXEC PGM=ENBPIU00,PARM='PARMLIST'

//PARMLIST DD \*

MODEL TBLOUT OPTIONS0 0

MODEL TBLOUT OPTIONS A

//POSITION DD \*

*The fields in the Endevor VALIDATE summary are defined*

NDVR#RC 40 43

Element 22 30

Environment 46 53

System 61 68

Subsystem 71 78

Type 83 90

Stgid 94 94

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

// DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINP.LOD

*The summary from the VALIDATE step*

//TABLE DD DSN=&&C1MSGS2,DISP=(OLD,DELETE)

//OPTIONS0 DD \*

FoundCount = 0

title = ' '

//MODEL DD \*

*CSV utility requests are output.*

\* Ele=&Element Env=&Environment Sys=&System Sub=&Subsystem Typ=&Type

LIST ELEMENT &Element FROM ENVIRONMENT &Environment STAGE &Stgid

SYSTEM &System SUBSYSTEM &Subsystem TYPE '&Type'

OPTIONS SEARCH RETURN FIRST &title

TO FILE APIEXTR .

//OPTIONS DD \*

*Only elements with the ‘0008’ return code are selected.*

***The CSV heading is requested only for the first element***

iF NDVR#RC /= '0008' then $SkipRow = 'Y'

**IF FoundCount > 0 THEN title = 'NOTITLE '**

FoundCount = 1

//SYSTSPRT DD SYSOUT=\*

//TBLOUTX DD SYSOUT=\*

//\*

*CSV requests*

//TBLOUT DD DSN=&&CSVLISTS,DISP=(NEW,PASS),

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=08000)

//\*--------------------------------------------------------------------\*

//\* STEP 2 -- EXECUTE CSV UTILITY

//\*--------------------------------------------------------------------\*

The Endevor CSV utility executes the constructed requests

//RETROFT4 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

*CSV requests are executed*

//BSTIPT01 DD DSN=&&CSVLISTS,DISP=(OLD,DELETE)

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//APIEXTR DD DSN=&&CSVFILE,

// DCB=(RECFM=FB,LRECL=1800,BLKSIZE=9000,DSORG=PS),

// DISP=(MOD,PASS),

// SPACE=(TRK,(5,1),RLSE)

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

//\*--------------------------------------------------------------------

Show the intermediate results

//SHOWME EXEC PGM=IEBGENER,REGION=1024K

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&CSVFILE,DISP=(OLD,PASS)

//SYSUT2 DD SYSOUT=\* OUTPUT FILE

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

//RETROFT5 EXEC PGM=ENBPIU00,

Table tool executed as compiled REXX

// PARM='M &SAVE#LIB &MRGE#LIB'

//TABLE DD \*

\* SaveLibrary PDMMergeLibrary

\* \*

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

//MODEL DD \*

*Saves JCL parameter values as OPTIONS for a subsequent step*

SaveLibrary = '&SaveLibrary'

PDMMergeLibrary = '&PDMMergeLibrary'

//OPTIONS DD DUMMY

//SYSTSPRT DD SYSOUT=\*

//TBLOUT DD DSN=&&OPTIONS,DISP=(NEW,PASS),

// UNIT=SYSDA,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=08000)

//\*--------------------------------------------------------------------\*

//RETROFT6 EXEC PGM=ENBPIU00,

Table tool executed as compiled REXX

// PARM='M &PDM#WIP &PENVIRON &PSTGID'

//TABLE DD \*

\* PDMWipLibrary ProdEnv ProdStgid

\* \* \*

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

//MODEL DD \*

PDMWipLibrary = '&PDMWipLibrary'

*Saves JCL parameter values as OPTIONS for a subsequent step*

ProdEnv = '&ProdEnv'

ProdStgid = '&ProdStgid'

//OPTIONS DD DUMMY

//SYSTSPRT DD SYSOUT=\*

//TBLOUT DD DSN=&&OPTIONS,DISP=(MOD,PASS)

//\*-------------------------------------------------------------------\*

Table tool executed as compiled REXX

//**RETROFT7** EXEC PGM=ENBPIU00,PARM='A'

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

*CSV output*

//TABLE DD DSN=&&CSVFILE,DISP=(OLD,DELETE)

**//MODEL DD DISP=SHR,DSN=SW.ENDV.SYA.ENDEVOR.MODELS(RETROFIX)**

*The MODEL is defined in a separate file (shown below)*

//OPTIONS DD DSN=&&OPTIONS,DISP=(OLD,DELETE)

// DD \*

Userid = USERID()

*Build the last character for a submitted job name, rotating the variable* ***jobchar*** *through the @, # and $ characters*

\* Create a last character for jobname:

remainder = $row# // 3 /\* Rexx Modular function \*/

**jobchar = Substr('@#$',(remainder+1),1)**

*$delimiter names a* ***character*** *to separate MODEL variables*

$delimiter = **'/'**

//SYSTSPRT DD SYSOUT=\*

//TBLOUTX DD SYSOUT=\*

*Output is written to the internal reader. A job is submitted for each element to be processed*

//TBLOUT DD SYSOUT=(A,INTRDR),LRECL=80

//

**The MODEL looks like this:**

*$delimiter names a* ***character*** *to separate MODEL variables*

//&Userid**/**&jobchar JOB (2,99539,FMCC-CSCS),'ENDEVOR',NOTIFY=&SYSUID,

// MSGCLASS=9,REGION=0M,TIME=1440,LINES=(9999,WARNING)

//\*-------------------------------------------------------------------\*

// SET ENVIRON=&ENV\_NAME

// SET SYSTEM=&SYS\_NAME

*Set statements at the top show reflect information for one element.*

*Values for variables in the MODEL are assigned during the* ***RETROFT7*** *step.*

// SET SUBSYS=&SBS\_NAME

// SET ELEMENT=&ELM\_NAME

// SET TYPE=&TYPE\_NAME

// SET STGID=&STG\_ID

// SET SAVE#LIB='&SaveLibrary'

// SET MRGE#LIB='&PDMMergeLibrary'

// SET PDM#WIP='&PDMWipLibrary'

//\*--------------------------------------------------------------------\*

Batch Endevor

//RETRIEVE EXEC PGM=NDVRC1,

// DYNAMNBR=1500,

// REGION=4096K,

// PARM='C1BM3000'

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//SAVE#LIB DD DISP=SHR,DSN=&SaveLibrary

//BSTIPT01 DD \*

RETRIEVE ELEMENT "&ELM\_NAME"

*Retrieve the element to the named ‘Save’ library*

FROM ENVIRONMENT "&ENV\_NAME"

SYSTEM "&SYS\_NAME" SUBSYSTEM "&SBS\_NAME"

TYPE "&TYPE\_NAME" STAGE "&STG\_ID"

TO DDNAME 'SAVE#LIB'

OPTIONS OVERRIDE SIGNOUT NO SIGNOUT

REPLACE

CCID &LAST\_ACT\_CCID COMMENT '&LAST\_ACT\_COMMENT'

.

//C1MSGS1 DD SYSOUT=\*

//C1MSGS2 DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//SYSTSPRT DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

//PDM EXEC PGM=NDVRC1,DYNAMNBR=1500,REGION=4096K,

Batch PDM execution

// PARM='BC1G0000'

//STEPLIB DD DISP=SHR,DSN=SDPROD.CAENDV.A.DEV.CSIQAUTU

//SYSEXEC DD DISP=SHR,DSN=SYS1.SISPEXEC

// DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.REX

// DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINP.REX

//SYSTSPRT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//C1TPDD01 DD UNIT=SYSDA,

// SPACE=(CYL,(1,3)),

// DISP=(NEW,DELETE,DELETE),

// DCB=(RECFM=VB,LRECL=260)

//C1TPDD02 DD UNIT=SYSDA,

// SPACE=(CYL,(1,3)),

// DISP=(NEW,DELETE,DELETE),

// DCB=(RECFM=VB,LRECL=260)

//C1TPLSIN DD UNIT=SYSDA,

// SPACE=(CYL,(1,3)),

// DISP=(NEW,DELETE,DELETE),

// DCB=(RECFM=FB,LRECL=80)

//C1TPLSOU DD UNIT=SYSDA,

// SPACE=(CYL,(1,3)),

// DISP=(NEW,DELETE,DELETE)

//C1PLMSGS DD SYSOUT=\*

//\* OUTPUT MESSAGE DATASETS

//C1MSGS1 DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSUDUMP DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

//\* PDM BATCH REQUEST DATASET

//BATCHIN DD \*

\* &ELM\_NAME &ENV\_NAME &SYS\_NAME &SBS\_NAME &TYPE\_NAME

*Execute PDM* ***Build WIP*** *and* ***MergeWIP*** *functions. Write the Merged results* ***to the output file***

**BUILD WIP** DSNAME '&PDMWipLibrary'

ROOT ENVIRONMENT &ENV\_NAME SYSTEM &SYS\_NAME SUBSYSTEM &SBS\_NAME

TYPE &TYPE\_NAME STAGE &STG\_ID

DV1 ENVIRONMENT &ENV\_NAME SYSTEM &SYS\_NAME SUBSYSTEM &SBS\_NAME

TYPE &TYPE\_NAME STAGE &STG\_ID

DV2 ENVIRONMENT &ProdEnv SYSTEM &SYS\_NAME SUBSYSTEM &SBS\_NAME

TYPE &TYPE\_NAME STAGE &ProdStgid

COMPARE 00001 THRU 00072

REPLACE .

WIP ' '

ROOT '&ELM\_NAME ' VER 01 LEV &BASE\_LL

DV1 '&ELM\_NAME '

DV2 '&ELM\_NAME ' .

**MERGE WIP** DSN '&PDMWipLibrary'

**OUTPUT DSN '&PDMMergeLibrary'**

REPLACE .

WIP &ELM\_NAME

OUT &ELM\_NAME

.

//\*--------------------------------------------------------------------\*

Table tool executed as compiled REXX

//RESULTS EXEC PGM=ENBPIU00,PARM='30'

//STEPLIB DD DISP=SHR,DSN=SW.ENDV.SYA.ADMINT.LOD

//TABLE DD DISP=SHR,DSN=&PDMWipLibrary(&ELM\_NAME)

*Only the first 30 records are scanned*

//POSITION DD \*

ConflictLiteral 04 22

*The WIP file member is scanned for the line that indicates the number of Conflicts*

ConflictNumber 37 38

//OPTIONS DD \*

$Table\_Type = "positions"

$TableHeadingChar = "|"

$TableCommentPrefix = "||"

If ConflictLiteral /= 'NUMBER OF CONFLICTS' then $SkipRow = 'Y'

*The return code for this step is made to match the number of Conflicts found.*

$my\_rc = ConflictNumber

TBLOUT = 'CONFLICT'

//MODEL DD \*

&ConflictLiteral = &ConflictNumber

//SYSTSPRT DD SYSOUT=\*

//CONFLICT DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

*Processing stops if any Conflicts are found*

//\*--------------------------------------------------------------------\*

//\*--------------------------------------------------------------------\*

//DELETE EXEC PGM=NDVRC1,COND=(0,NE,RESULTS),

Batch Endevor step

// DYNAMNBR=1500,

// REGION=4096K,

// PARM='C1BM3000'

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//BSTIPT01 DD \*

DELETE ELEMENT "&ELM\_NAME"

*Delete the element*

FROM ENVIRONMENT "&ENV\_NAME"

SYSTEM "&SYS\_NAME" SUBSYSTEM "&SBS\_NAME"

TYPE "&TYPE\_NAME" STAGE "&STG\_ID"

OPTIONS OVERRIDE SIGNOUT

CCID &LAST\_ACT\_CCID COMMENT '&LAST\_ACT\_COMMENT'

.

//SYSTSPRT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//C1MSGS1 DD SYSOUT=\*

//C1MSGS2 DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

Batch Endevor step

//ADD#UPD1 EXEC PGM=NDVRC1,COND=(0,NE,RESULTS),

// DYNAMNBR=1500,

// REGION=4096K,

// PARM='C1BM3000'

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//SAVE#LIB DD DISP=SHR,DSN=&SaveLibrary

//BSTIPT01 DD \*

ADD ELEMENT "&ELM\_NAME"

TO ENVIRONMENT "&ENV\_NAME"

SYSTEM "&SYS\_NAME" SUBSYSTEM "&SBS\_NAME"

*Add the element back from the* ***Saved source***

TYPE "&TYPE\_NAME"

**FROM DDNAME 'SAVE#LIB'**

OPTIONS OVERRIDE SIGNOUT

CCID &LAST\_ACT\_CCID COMMENT '&LAST\_ACT\_COMMENT'

UPDATE IF PRESENT

BYPASS GENERATE PROCESSOR

.

//SYSTSPRT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//C1MSGS1 DD SYSOUT=\*

//C1MSGS2 DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*--------------------------------------------------------------------\*

//ADD#UPD2 EXEC PGM=NDVRC1,COND=(0,NE,RESULTS),

Batch Endevor step

// DYNAMNBR=1500,

// REGION=4096K,

// PARM='C1BM3000'

//SYSEXEC DD DSN=SYS1.SISPEXEC,DISP=SHR

//MRGE#LIB DD DISP=SHR,DSN=&PDMMergeLibrary

//BSTIPT01 DD \*

ADD ELEMENT "&ELM\_NAME"

TO ENVIRONMENT "&ENV\_NAME"

*Update the element from the* ***Merged*** *source*

SYSTEM "&SYS\_NAME" SUBSYSTEM "&SBS\_NAME"

TYPE "&TYPE\_NAME"

**FROM DDNAME 'MRGE#LIB'**

OPTIONS OVERRIDE SIGNOUT

CCID &LAST\_ACT\_CCID COMMENT '&LAST\_ACT\_COMMENT'

**UPDATE IF PRESENT**

BYPASS GENERATE PROCESSOR

.

//SYSTSPRT DD SYSOUT=\*

//SYSTSIN DD DUMMY

//C1MSGS1 DD SYSOUT=\*

//C1MSGS2 DD SYSOUT=\*

//SYSUDUMP DD SYSOUT=\*

//SYMDUMP DD DUMMY

//SYSOUT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

# Example #7. Convert PDS datasets into PDS/E datasets.

Featuring:

1. Using a a PARMLIST to produce multiple kinds of output
2. A POSITION file to define locations and lengths of fields within the TABLE
3. Generations of multiple IEBCOPY commands to run in a single step

//\*--------------------------------------------------------------------\*

//\* STEP 1 – Read the saved 3.4 list of datasets

//\*--------------------------------------------------------------------\*

*The Table Tool (ENBPIU00) executed as interpreted REXX.*

//BUILDS EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

//SYSEXEC DD DISP=SHR,DSN=&SYSEXEC

//TABLE DD DISP=SHR,DSN=&DSNLIST

*This dataset is created by:*

* *Using TSO option 3.4 to list datasets*
* *Entering “Save” and a name on the command line*
* *TSO creates a dataset named <yourUserid>.<name>.DATASETS.*

//PARMLIST DD \*

MODEL1 TBLOUT1 OPTIONS A

MODEL2 TBLOUT2 OPTIONS A

//MODEL1 DD \*

ALLOC F(INDD) +

DA('&Dataset') SHR REUSE

ALLOC F(OUTDD) +

*Model1 provides format for 1 kind of output*

DA('&Dataset.NEW') +

NEW LIKE ('&Dataset') +

DSNTYPE(LIBRARY)

CALL \*(IEBCOPY)

FREE F(INDD)

FREE F(OUTDD)

//MODEL2 DD \*

*Model2 provides format for a 2nd kind of output*

RENAME '&Dataset' +

'&Dataset.OLD'

RENAME '&Dataset.NEW' +

'&Dataset'

//OPTIONS DD DUMMY

*Field positions in the SAVEd dataset*

//POSITION DD \*

Dataset 01 44

*Field positions in the SAVEd dataset*

Dsorg 53 54

Tracks 80 88

CreateDate 108 117

AccessDate 130 139

//SYSTSPRT DD SYSOUT=\*

//TBLOUT1X DD SYSOUT=\*

//TBLOUT2X DD SYSOUT=\*

//\*

//TBLOUT1 DD DSN=&&TBLOUT1,DISP=(NEW,PASS),

*Output for the ALLOCS step*

// SPACE=(CYL,(1,1)),UNIT=SYSDA,

// LRECL=80,RECFM=FB,BLKSIZE=0

//TBLOUT2 DD DSN=&&TBLOUT2,DISP=(NEW,PASS),

*Output for the RENAMES step*

// SPACE=(CYL,(1,1)),UNIT=SYSDA,

// LRECL=80,RECFM=FB,BLKSIZE=0

//\*--------------------------------------------------------------------

*After each PDS is allocated as INDD and a new PDS/E is allocated as OUTDD, IEBCOPY is called.*

//**ALLOCS** EXEC PGM=IKJEFT1B

//SYSTSIN DD DSN=&&TBLOUT1,DISP=(OLD,DELETE)

//SYSIN DD \*

*Each IEBCOPY execution uses the same SYSIN which is defined only once here.*

COPY INDD=INDD,OUTDD=OUTDD

//SYSUT2 DD UNIT=SYSDA,SPACE=(CYL,(90,90))

//SYSUT3 DD UNIT=SYSDA,SPACE=(CYL,(90,90))

//SYSUT4 DD UNIT=SYSDA,SPACE=(CYL,(90,90))

//SYSUT5 DD UNIT=SYSDA,SPACE=(CYL,(90,90))

//SYSUT6 DD UNIT=SYSDA,SPACE=(CYL,(90,90))

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*--------------------------------------------------------------------

*PDS files are renamed from filename to filename.OLD.*

*PDS/E files named filename.NEW are renamed as filename*

//**RENAMES** EXEC PGM=IKJEFT1B

//SYSTSIN DD DSN=&&TBLOUT2,DISP=(OLD,DELETE)

//SYSTSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//\*--------------------------------------------------------------------

# Other example uses of Table Tool:

1. Endevor’s Best Practice Implementation (BPI) (See the Endevor **Scenario Guide** )
   1. Tables for Environments, Systems, Subsystems, Types, processor groups and files
   2. Table Tool to construct and Endevor tables including C1DEFLTS and ESYMBOLS
   3. Table Tool execution to define all Systems, Subsystems, Types, processor groups
   4. Table Tool execution to allocate all libraries and VSAM files
   5. Load sample inventory
2. Parse the Endevor Component list within a MOVE processor to build BSTCOPY statements
3. Management of long-named USS directories (larger than 70 bytes) within a processor
4. ENBPIU00 Called from exit programs (can run foreground or batch without TSO)
5. Bundles