**Table Tool Examples**

Prepared for:

CA Endevor Community

August , 2022

Document Properties

|  |  |
| --- | --- |
| Attribute | Value |
| Document Name: | Table Tool Examples 2020 03 |
| Document Version: |  |
| Version Date: | August 2022 |
| Document Status: |  |
| Authors | Dan Walther |

Copyright © 2019 Broadcom Inc. All rights reserved. All trademarks, trade names, service marks and logos referenced herein belong to their respective companies.

This document is for your informational purposes only. To the extent permitted by applicable law, Broadcom provides this document ‘AS IS’ without warranty of any kind, including, without limitation, any implied warranties of merchantability or fitness for a particular purpose, or non-infringement. In no event will Broadcom be liable for any loss or damage, direct or indirect, from the use of this document including, without limitation, lost profits, business interruption, goodwill or lost data, even if Broadcom is expressly advised of such damages.

Document Overview

| Item | Description |
| --- | --- |
| Document Objective: | This document offers examples for the use of the “Table Tool”, which is available at every Endevor site. |
| Audience: | Primary: Endevor AdministratorsSecondary: Mainframe users |

Review & Approval

| Review Date | Action | Name | Company, Organizational Position |
| --- | --- | --- | --- |
|  | Author | Dan Walther | Broadcom |
|  |  |  |  |

Table Tool Examples

Contents

[Example #1. Identify and DELETE old elements in a development environment. 4](#_Toc111986001)

[Example #2a. Build a report of Package information using CSV 7](#_Toc111986002)

[Example #2b. Build a report of Package information – without CSV 10](#_Toc111986003)

[Example #3. Calculate and report information about datasets listed in a saved 3.4 dataset list. 14](#_Toc111986004)

[Example #4. Build Add statements for members of a PDS, where the members may already exist in Endevor with various System, Subsystem and Type names. 17](#_Toc111986005)

[Example #5. Convert PDS datasets into PDS/E datasets. 21](#_Toc111986006)

[Example #6. Report Endevor processor usage, and list of processor elements that are not used anywhere. 23](#_Toc111986007)

[Example #7. Table Tool within a JCLCheck processor for robust ACM data collection 32](#_Toc111986008)

[Other example uses of Table Tool: 38](#_Toc111986009)

# Example #1. 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 the* ***Reporting*** *section of the* ***TechDocs***  *documentation for Endevor.*

*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 #2a. 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 the* ***Reporting*** *section of the* ***TechDocs***  *documentation for Endevor.*

*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 #2b. 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 the* ***Reporting*** *section of the* ***TechDocs***  *documentation for Endevor.*

*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 #3. 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 #4. 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 #5. 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
4. Can be made to run with BSTXCOPY. Can use Rexx statements with Table Tool to enlarge new datasets based on a factor to be applied to existing dataset sizes

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

//\* 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=\*

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

# Example #6. Report Endevor processor usage, and list of processor elements that are not used anywhere.

Featuring:

1. Multiple Endevor CSV steps to collect Endevor information
2. Table Tool steps that process CSV data
3. Leverage of the REXX stem array feature for randomly storing and retrieving of data.

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

// SET WORKLIB=WALJO11.PARMS

*Name a dataset for intermediate and final results*

// SET CSIQCLS0=CAPRD.NDVR.PROD.CATSNDVR.CSIQCLS0

// SET NDVR#HLQ=CAPRD.NDVR.V180CA06

*Name your CSIQCLS0 and you’re the HLQ for your Endevor CSIQAUTU, CSIQAUTH and CSIQLOAD datasets*

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

//\* Report processor usage

//\* Outputs:

//\* &WORKLIB(PRC#ENV1) WALJO11.PARMS(PRC#ENV1)

//\* &WORKLIB(PRC#ENV2) WALJO11.PARMS(PRC#ENV2)

//\* &WORKLIB(PRC#GRPS) WALJO11.PARMS(PRC#GRPS)

//\* &WORKLIB(PROCUSED) WALJO11.PARMS(PROCUSED)

//\* &WORKLIB(NOTUSED)) WALJO11.PARMS(NOTUSED))

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

//\* STEP 1 -- EXECUTE CSV UTILITY to Gather Environment info

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

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

*This step executes a CSV to identify your environments*

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTU SCMM@LIB

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTH SCMM@LIB

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQLOAD SCMM@LIB

//CONLIB DD DISP=SHR,DSN=&NDVR#HLQ..CSIQLOAD SCMM@LIB

//BSTIPT01 DD \*

LIST ENVIRONMENT

'\*'

 TO DDNAME 'EXTRACTS'

 OPTIONS RETURN ALL.

//EXTRACTS DD DSN=&&EXTRACTS,

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

// DISP=(NEW,PASS),

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

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

//\*------

//WORKLIB DD DSN=&WORKLIB,

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=0,DSORG=PO),

// DISP=(MOD,CATLG,KEEP),DSNTYPE=LIBRARY,

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

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

//\*------ Reformat Envionments into CSV requests for Procesor Grp info-

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

*Table Tool reads the CSV output and for each environment creates outputs for subsequent steps:*

//STEP1A EXEC PGM=IRXJCL,PARM='ENBPIU00 A'

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

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

//MODEL1 DD \*

 LIST PROCESSOR GROUP '\*'

 FROM ENVIRONMENT '&ENV\_NAME' STAGE '\*'

*A CSV request to collect Processor group information*

 SYSTEM '\*' TYPE '\*'

 OPTIONS NOSEARCH RETURN ALL TO FILE CSVEXTR .

//MODEL2 DD \*

LIST ELEMENT

'\*'

*Element CSV requests to List all processor elements*

 FROM ENVIRONMENT &ENV\_NAME SYSTEM '\*' SUBSYSTEM '\*'

 TYPE 'PROCESS'

 STAGE NUMBER '\*'

 TO DDNAME 'LISTELMS'

 OPTIONS SEARCH RETURN FIRST PATH PHYSICAL .

//OPTIONS DD \* CONTROL STATEMENTS

 $NumberModelsAndTblouts= 2

//SYSTSPRT DD SYSOUT=\*

//TBLOUT1 DD DISP=SHR,DSN=&WORKLIB(PRC#ENV1)

//TBLOUT2 DD DISP=SHR,DSN=&WORKLIB(PRC#ENV2)

//\*

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

//\* STEP 2 -- EXECUTE CSV UTILITY to collect processor group info

*The CSV utility executes the LIST PROCESSOR GROUP commands created by Table Tool*

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

//STEP2 EXEC PGM=NDVRC1,REGION=4M,

// PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTU SCMM@LIB

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTH SCMM@LIB

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQLOAD SCMM@LIB

//CONLIB DD DISP=SHR,DSN=&NDVR#HLQ..CSIQLOAD SCMM@LIB

//BSTIPT01 DD DISP=SHR,DSN=&WORKLIB(PRC#ENV1)

//CSVEXTR DD DSN=&&CSVFILE,

// DCB=(RECFM=VB,LRECL=4092,BLKSIZE=4096,DSORG=PS),

// DISP=(MOD,PASS),

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

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

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

//\*------ Analyze Processor Group CSV data

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

*This Table Tool Step analyzes the Processor Group data. It builds a report of processor usage, and marks processors that are defined in at least one processor group.*

//STEP2A EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

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

//PARMLIST DD \*

 NOTHING NOTHING OPTIONS0 0

 NOTHING NOTHING OPTIONS1 A

 HEADING TBLOUT1 NOTHING 1

 MODEL TBLOUT2 OPTIONS2 A

 IAMUSED PROCUSED OPTIONS3 A

//HEADING DD \*

*This is the heading placed at the top of the Processor usage report.*

\*Processr Environ- S System-- Type---- PrcGrp-- ProcType

//MODEL DD \*

*This is the MODEL for detail lines of the Processor usage report*

 &PROC\_NAME &ENV\_NAME &STG\_ID &SYS\_NAME &TYPE\_NAME &PROC\_GRP\_NAME &PROC\_TYPE

//OPTIONS0 DD \* CONTROL STATEMENTS

 Processor. = ''

 ProcessorUsed. = 'N'

 $nomessages = 'Y' /\* Bypass messages Y/N \*/

//OPTIONS1 DD \* CONTROL STATEMENTS

 if PROC\_NAME = '\*NOPROC\*' Then $SkipRow = 'Y'

 entry= ENV\_NAME SYS\_NAME TYPE\_NAME STG\_ID PROC\_TYPE PROC\_GRP\_NAME

 entry = Translate(entry,'\_',' ')

 PROC\_NAME = Strip(PROC\_NAME) ;

 If Wordpos(entry,Processor.PROC\_NAME) = 0 then, +

 Processor.PROC\_NAME = Processor.PROC\_NAME entry ;

 ProcessorUsed.PROC\_NAME = 'Y'

 $SkipRow = 'Y'

//IAMUSED DD \*

*REXX Stem arrays are created here, and used in subsequent steps. The REXX statements are written to the WORKLIB as member PROCUSED.*

 ProcessorUsed.&PROC\_NAME = 'Y'

//PROCUSED DD DISP=SHR,DSN=&WORKLIB(PROCUSED)

//NOTHING DD DUMMY CONTROL STATEMENTS

//OPTIONS2 DD \* CONTROL STATEMENTS

 PROC\_NAME = Strip(PROC\_NAME) ;

 if PROC\_NAME = '\*NOPROC\*' Then $SkipRow = 'Y'

 ProcessorEntries = Processor.PROC\_NAME ;

 Processor.PROC\_NAME = '' ;

 If ProcessorEntries = '' then $SkipRow = 'Y'

 PROC\_NAME = Left(PROC\_NAME,8) ;

 Do ent# = 1 to Words(ProcessorEntries); +

 entry= Word(ProcessorEntries,ent#) ; +

 entry = Translate(entry,' ','\_') ; +

 ENV\_NAME = Left(Word(entry,1),8); +

 SYS\_NAME = Left(Word(entry,2),8); +

 TYPE\_NAME = Left(Word(entry,3),8); +

 STG\_ID = Left(Word(entry,4),1); +

 PROC\_TYPE = Left(Word(entry,5),8); +

 PROC\_GRP\_NAME = Left(Word(entry,6),8); +

 x = BuildFromModel(MODEL); +

 Processor.PROC\_NAME = '' ;+

 End;

 $SkipRow = 'Y'

//OPTIONS3 DD \* CONTROL STATEMENTS

 PROC\_NAME = Strip(PROC\_NAME);

 If PROC\_NAME = '\*NOPROC\*' then $SkipRow = 'Y' ;

 IsProcessorUsed = ProcessorUsed.PROC\_NAME

 If IsProcessorUsed = 'Y' then x = BuildFromModel(IAMUSED)

 ProcessorUsed.PROC\_NAME = 'done'

 $SkipRow = 'Y'

//SYSTSPRT DD SYSOUT=\*

//NOTHING DD DUMMY CONTROL STATEMENTS

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

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

// UNIT=SYSALLDA,SPACE=(CYL,(1,5),RLSE),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=0,DSORG=PS)

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

// UNIT=SYSALLDA,SPACE=(CYL,(1,5),RLSE),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=0,DSORG=PS)

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

//\* SORT DATA \*

*The Processor Group data is sorted before being placed into the WORKLIB as member PRC#GRPS.*

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

//SORT EXEC PGM=SORT

//SYSPRT DD SYSOUT=\*

//SYSPRINT DD SYSOUT=\*

//SYSOUT DD SYSOUT=\*

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

//SORTOUT DD DSN=&&TBLOUT2A,DISP=(NEW,PASS),

// UNIT=SYSALLDA,SPACE=(CYL,(1,5),RLSE),

// DCB=(RECFM=FB,LRECL=080,BLKSIZE=0,DSORG=PS)

//SYSIN DD \*

 SORT FIELDS=(01,80,CH,A)

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

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

//SYSPRINT DD SYSOUT=\* MESSAGES

//SYSUT1 DD DSN=&&TBLOUT1,DISP=(OLD,PASS,DELETE)

// DD DSN=&&TBLOUT2A,DISP=(OLD,PASS,DELETE)

//SYSIN DD DUMMY CONTROL STATEMENTS

//SYSUDUMP DD SYSOUT=\*

//SYSUT2 DD DISP=SHR,DSN=&WORKLIB(PRC#GRPS)

//\*

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

//\*------ Report UnUsed processors.

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

//\* STEP 3 -- EXECUTE CSV UTILITY

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

*This step executes the Element CSV requests to List all processor elements*

//STEP3 EXEC PGM=NDVRC1,REGION=4M,/ PARM='BC1PCSV0'

//STEPLIB DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTU

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQAUTH

// DD DISP=SHR,DSN=&NDVR#HLQ..CSIQLOAD

//BSTIPT01 DD DISP=SHR,DSN=&WORKLIB(PRC#ENV2)

//LISTELMS DD DSN=&&LISTELMS,

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

// DISP=(MOD,PASS),

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

//C1MSGS1 DD SYSOUT=\*

//BSTERR DD SYSOUT=\*

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

//\*------ Report UnUsed processors.

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

*This Table Tool step merges the list of processor elements with processors defined in processor groups*

//STEP3A EXEC PGM=IRXJCL,PARM='ENBPIU00 PARMLIST'

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

//PARMLIST DD \*

 NOTHING NOTHING OPTIONS0 0

 HEADING NOTUSED PROCUSED 1

 MODEL NOTUSED OPTIONS2 A

//OPTIONS0 DD \*

 ProcessorUsed. = 'N'

*REXX Stem arrays are used here, and identify which processors are in-use.*

 $nomessages = 'Y' /\* Bypass messages Y/N \*/

//PROCUSED DD DISP=SHR,DSN=&WORKLIB(PROCUSED)

//HEADING DD \*

\*Element- Environ- System-- SubSystm Type---- St Msg------

//MODEL DD \*

 &ELM\_NAME &ENV\_NAME &SYS\_NAME &SBS\_NAME &TYPE\_NAME &STG\_ID not used

//OPTIONS2 DD \* CONTROL STATEMENTS

 element = Strip(ELM\_NAME)

 If Words(element) > 1 then $SkipRow = 'Y'

 IsProcessorUsed = ProcessorUsed.element

 If IsProcessorUsed /= 'N' then $SkipRow = 'Y'

 ProcessorUsed.element = 'done'

\* Format outputs

 ELM\_NAME = Left(ELM\_NAME,8)

 ENV\_NAME = Left(ENV\_NAME,8)

 SYS\_NAME = Left(SYS\_NAME,8)

 SBS\_NAME = Left(SBS\_NAME,8)

 TYPE\_NAME = Left(TYPE\_NAME,8)

//SYSTSPRT DD SYSOUT=\*

//NOTHING DD DUMMY

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

*A list of processor elements not defined on any processor group, is placed into the WORKLIB as member NOTUSED..*

//TBLOUT DD SYSOUT=\*

//NOTUSED DD DISP=SHR,DSN=&WORKLIB(NOTUSED)

**Sample output PRC#GRPS member:**

\*Processr Environ- S System-- Type---- PrcGrp-- ProcType

 COPY#LIB SMPLPROD E CATSNDVR CLIST LIVE GEN

- - - - - - - - - - - - - - - - 37 Line(s) not Displayed

 DELACMOP DEV 1 ALC CICSMAP MAPASM DEL

 DELACMOP DEV 1 DEVOPS CBL LOAD DEL

 DELACMOP DEV 1 DEVOPS CICSMAP MAPASM DEL

 DELACMOP DEV 1 FINANCE CBL LOAD DEL

 DELACMOP DEV 1 FINANCE CICSMAP MAPASM DEL

- - - - - - - - - - - - - - - - 27 Line(s) not Displayed

 DELETE SMPLPROD E CATSNDVR ASM TABLE DEL

 DELETE SMPLPROD E CATSNDVR CLIST LIVE DEL

 DELETE SMPLPROD E CATSNDVR CMDTABLE LIVE DEL

 DELETE SMPLPROD E CATSNDVR JCL LIVE DEL

- - - - - - - - - - - - - - - 482 Line(s) not Displayed

 ENXDSAND DEV 1 ALC SANDBTRG SANDBTRG DEL

 ENXDSAND DEV 2 ALC SANDBTRG SANDBTRG DEL

 ENXGSAND DEV 1 ALC SANDBTRG SANDBTRG GEN

 ENXGSAND DEV 2 ALC SANDBTRG SANDBTRG GEN

- - - - - - - - - - - - - - - - 17 Line(s) not Displayed

DELETE SMPLPROD E CATSNDVR ASM TABLE DEL

 DELETE SMPLPROD E CATSNDVR CLIST LIVE DEL

 DELETE SMPLPROD E CATSNDVR CMDTABLE LIVE DEL

 DELETE SMPLPROD E CATSNDVR JCL LIVE DEL

- - - - - - - - - - - - - - - 482 Line(s) not Displayed

 ENXDSAND DEV 1 ALC SANDBTRG SANDBTRG DEL

 ENXDSAND DEV 2 ALC SANDBTRG SANDBTRG DEL

 ENXGSAND DEV 1 ALC SANDBTRG SANDBTRG GEN

 ENXGSAND DEV 2 ALC SANDBTRG SANDBTRG GEN

- - - - - - - - - - - - - - - - 17 Line(s) not Displayed

 GASM DEV 1 DEVOPS ASMPGM LOAD GEN

 GASM DEV 1 DEVOPS ASMPGM LOADRN GEN

- - - - - - - - - - - - - - - 201 Line(s) not Displayed

 GCIINBL SMPLTEST T FINANCE COBOL C2BD#### GEN

 GCIINBL SMPLTEST T FINANCE COBOL C2BD#M## GEN

 GCIINBL SMPLTEST T FINANCE COBOL C2O##### GEN

 GCIINBL SMPLTEST T FINANCE COBOL C2O##M## GEN

 GCIINBL SMPLTEST T FINANCE COBOL C2OD#### GEN

- - - - - - - - - - - - - - - - 39 Line(s) not Displayed

 GCOB DEV 1 DEVOPS CBL LOAD GEN

 GCOB DEV 1 DEVOPS COBOL GOFDCOB1 GEN

- - - - - - - - - - - - - - - - 12 Line(s) not Displayed

 GCOB DEV 1 FINANCE COBOL LOAD1 GEN

 GCOB DEV 1 FINANCE COBOL LOAD2 GEN

- - - - - - - - - - - - - - - - 86 Line(s) not Displayed

 GCOBGIB PRD 1 DEVOPS COBOL LOADGIB GEN

 GCOBGIB PRD 1 FINANCE COBOL LOADGIB GEN

 GCOBGIB PRD 2 DEVOPS COBOL LOADGIB GEN

 GCOBGIB PRD 2 FINANCE COBOL LOADGIB GEN

 GCOBOL SMPLPROD E CATSNDVR COBOL COBOL GEN

 GCOBOL SMPLPROD E CATSNDVR COBOL EXIT GEN

 GCOBOL SMPLPROD E CONCURNT COBOL BATCH GEN

 GCOBOL SMPLPROD E CONCURNT COBOL ONLINE GEN

- - - - - - - - - - - - - - - - 48 Line(s) not Displayed

 GJAVA QA 1 DEVOPS JAVA JAVA GEN

 GJAVA QA 1 FINANCE JAVA JAVA GEN

 GJAVA QA 2 DEVOPS JAVA JAVA GEN

 GJAVA QA 2 FINANCE JAVA JAVA GEN

 GJAVA SMPLTEST Q CONCURNT JAVA JAVA GEN

 GJAVA SMPLTEST Q CONCURNT JAVA JAVAAPP GEN

**Sample output NOTUSED member:**

\*Element- Environ- System-- SubSystm Type---- St Msg------

 DB2COMP PRD EA PROCESS PROCESS 1 not used

 DELPROC PRD EA PROCESS PROCESS 1 not used

 DONTUSE PRD EA PROCESS PROCESS 1 not used

 ESPEVENT PRD EA PROCESS PROCESS 1 not used

 FOFDCOB PRD EA PROCESS PROCESS 1 not used

 GASMDBL1 PRD EA PROCESS PROCESS 1 not used

 GASMDB2 PRD EA PROCESS PROCESS 1 not used

 GASMNCLD PRD EA PROCESS PROCESS 1 not used

 GASMS10A PRD EA PROCESS PROCESS 1 not used

 GCIIDBL PRD EA PROCESS PROCESS 1 not used

 GCOBCICS PRD EA PROCESS PROCESS 1 not used

 GCOBDB2 PRD EA PROCESS PROCESS 1 not used

 GCOBIDMS PRD EA PROCESS PROCESS 1 not used

 GCOBW PRD EA PROCESS PROCESS 1 not used

 GCPYIMP PRD EA PROCESS PROCESS 1 not used

 GENPL1 PRD EA PROCESS PROCESS 1 not used

 GESPEVNT PRD EA PROCESS PROCESS 1 not used

 GESPSIM PRD EA PROCESS PROCESS 1 not used

 GENPL1 PRD EA PROCESS PROCESS 1 not used

 GFORTRAN PRD EA PROCESS PROCESS 1 not used

 GJAVA2 PRD EA PROCESS PROCESS 1 not used

 GJCLC PRD EA PROCESS PROCESS 1 not used

 GJCLPRD PRD EA PROCESS PROCESS 1 not used

 GLNKNBL PRD EA PROCESS PROCESS 1 not used

 GMAP PRD EA PROCESS PROCESS 1 not used

 GPDSMBR PRD EA PROCESS PROCESS 1 not used

 GPLI PRD EA PROCESS PROCESS 1 not used

 GPPASM PRD EA PROCESS PROCESS 1 not used

 GXML PRD EA PROCESS PROCESS 1 not used

 JDGCBDB2 PRD EA PROCESS PROCESS 1 not used

 MBPLOAD PRD EA PROCESS PROCESS 1 not used

 MJAVA2 PRD EA PROCESS PROCESS 1 not used

# Example #7. Table Tool within a JCLCheck processor for robust ACM data collection

Featuring:

1. Using Table Tool within a processor that executes JCLCheck and leverages its XREF data feature. ACM relationships between the JCL and other objects, such as PROCs, Programs, and datasets are collected. See the JCLCheck XREF feature described here.

<https://techdocs.broadcom.com/content/broadcom/techdocs/us/en/ca-mainframe-software/automation/ca-jclcheck-workload-automation/12-0/using/use-ca-jclcheck/xref-reporting-using-the-updtdsn-updtdd-options.html>

1. A POSITION file to define locations and lengths of fields within the TABLE
2. CONRELE to save ACM relationships captured by the JCLCheck XREF

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

//\*\* GJCL JCLCHECK processor \*\*

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

//JCLGCA7C PROC AAAAAA=,

// CNTLCRDS=&HLQ01.&C1ENVMNT..&C1STAGE..PARMLIB.APP,

*Endevor processor, with processor symbols at the top..*

// ETYP='&C1PRGRP(4,1)', T/Q 4'th char of processor grp

// EUID='&C1ELEMENT(1,5)', ELEMENT USERID

// EXPINC=N, EXPAND INCLUDES

// HLQ01='ENDPCPB.N.', ENDEVOR CENTRAL

// JCL#OPT1=' CC(5) CT SP(RPT 65 NOALL) ',

// JCL#OPT2=' V PROC(SYSPROC) ',

// JCL#OPT3=' F SX RP ER( END) ',

// JCL#OPT4=' SYN RUNT J ',

// JCL#OPT5=' NOAUD NOAUTOP ',

// JCL#OPT6=' NOPDS ',

// LASTNODE=&LASTND@&ETYP, Last node for datasets

// LASTND@T='CA7QTST', for prgrp ending T

// LASTND@Q='CA7QQA', for prgrp ending Q

// OUTLIB='SYSDE32.NDVR.&&C1ENVMNT.&&C1SYSTEM.&&C1SU.&LASTNODE',

// PARM='OPTIONS(JCLOPTS)',

// PRCLIB='SYSDE32.NDVR.&&C1ENVMNT.&&C1SYSTEM.&&C1SU.PROC',

// LSTLIB='SYSDE32.NDVR.&&C1ENVMNT.&&C1SYSTEM.&&C1SU.LISTLIB',

// PRDLIB='', PNCPPPB.SUBLIB..&LASTNODE

// SHOWME='Y', Show diagnostics & intermediate results Y/N

// QAST='QAST',

// STDTABLE='MVSPPPS.N.JCLCHECK.TABLES(JCKTBLE)',

// SYSEXEC='TSOPPPE.N.COMMON.CLIST',

// ZZZZZZZ= the end

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

//\* COPY SOURCE TO ENDEVOR LIBRARY \*

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

//WRITEEND EXEC PGM=CONWRITE, < Save JCL into a PDS

// PARM='EXPINCL(&EXPINC)',

*Processor step to write the JCL element to an output lib*

// COND=(0,LT),MAXRC=0

//ELMOUT DD DSN=&OUTLIB(&C1ELEMENT),

// DISP=SHR,MONITOR=COMPONENTS

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

//\* ALLOCATE LISTING DATASETS \*

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

*Processor step to allocate temporary output datasets*

//ALLOCLST EXEC PGM=BC1PDSIN, < Allocate outputs

// MAXRC=0

//C1INIT01 DD DSN=&&LIST01,DISP=(,PASS),

// UNIT=3390,SPACE=(TRK,(1,5)),

// DCB=(RECFM=FBA,LRECL=133,BLKSIZE=13300,DSORG=PS)

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

//\* CA JCLCHECK \*

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

//JCLCHECK EXEC PGM=JCLCHECK, < Run JCLCheck to validate JCL

Processor Step to execute JCLCHECK

// PARM='&PARM',

// ALTID=N,

*Optionally specify the step is to run under the Alt id*

// COND=(0,LT),MAXRC=4

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

//SYSTSPRT DD SYSOUT=\* REXX MESSAGES

//SYSPRINT DD DSN=&&LIST01,DISP=(OLD,PASS)

//\* \*\*\*\* JCLCHECK PROC LIBRARIES \*\*\*\*

//SYSPROC DD DISP=(,DELETE),DSN=&&SYSPROC,

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

*MONITOR=COMPONENTS is used to collect JCL relationships with PROCs*

J

// DSORG=PO,RECFM=FB,LRECL=80,BLKSIZE=27920

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

// ALLOC=(LMAP,COND),MONITOR=COMPONENTS

// DD DISP=SHR,DSN=SYSP.PROCLIB

// ALLOC=COND,MONITOR=COMPONENTS

// DD DISP=SHR,DSN=SYS1.PRODUSER,

*ALLOC=LMAP allows Endevor to use the map for including multiple datasets*

*ALLOC=COND is used in case the dataset does not exist on this site.*

// ALLOC=COND,MONITOR=COMPONENTS

//\* \*\*\*\* CONTROL CARDS LIBRARIES \*\*\*\*

//\*

//\*CNTLCRDS DD DISP=SHR,DSN=&CNTLCRDS,

//\* ALLOC=(LMAP,COND),MONITOR=COMPONENTS

//\* DD DISP=SHR,DSN=&HLQ01.PRD.PROD.PARMLIB

//\*

//SYSIN DD DSN=&OUTLIB(&C1ELEMENT), INPUT JCL

// DISP=SHR

//JCLOPTS DD \*

&JCL#OPT1

&JCL#OPT2

&JCL#OPT3

&JCL#OPT4

&JCL#OPT5

&JCL#OPT6

JCLCHECK request for XREF data

 XREF PXREF UPDTDD(XREFUPDT)

//\*

//XREFUPDT DD DSN=&&XREFUPDT,DISP=(,PASS),

// DCB=(LRECL=1024,RECFM=FB,BLKSIZE=27648),

*XREF data is written here*

// UNIT=3390,SPACE=(CYL,(1,5))

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

//\* Create Endevor RELATE statements from JCLCHECK's XREFUPDT output \*

*Execute Table Tool to create Endevor “RELATE”s*

J

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

//RELATES1 EXEC PGM=IRXJCL, <-Create Relate stmts

// PARM='ENBPIU00 A'

*The JCLCheck XREF data is input*

//TABLE DD DSN=&&XREFUPDT,DISP=(OLD,PASS)

//POSITION DD DISP=SHR,DSN=SYSDE32.NDVR.TEAM.PARM(JCLCKXRF)

*The format of the XREF data is defined here*

//OPTIONS DD \*

 $Table\_Type = "positions"

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

 If XRF\_TYPE = 'DSN' & Substr(XRF\_DSN\_NAME,1,1) = '50'x then, +

 $SkipRow = 'Y'

 If XRF\_TYPE /= 'DSN' & XRF\_TYPE /= 'PGM' then $SkipRow = 'Y'

 $my\_rc = 1

 If XRF\_TYPE = 'PGM' then, +

 Do; +

 XRF\_PROC\_PROCLIB = Strip(XRF\_PROC\_PROCLIB) ; +

 XRF\_PROC\_PROCLIB = Translate(XRF\_PROC\_PROCLIB,'\*','7D'x); +

 XRF\_PROC\_PROCLIB = Translate(XRF\_PROC\_PROCLIB,'\*',' '); +

 entry = XRF\_PROC\_PROCLIB'('|| Strip(XRF\_DSN\_MEM) ||')'; +

 entry = Translate(entry,'\_',' ') ; +

 If Wordpos(entry,AlreadyDone) = 0 then, +

Rexx statements tell Table Tool which variables to use with each output Model.

Additional REXX checking prevent the construction of duplicate Relate statements.

 Do; +

 x= BuildFromMODEL(MODELPGM); +

 AlreadyDone = AlreadyDone entry; +

 End; +

 $SkipRow = 'Y'; +

 End

\*

 MODEL = 'MODELDSN'

 DSNInfo = XRF\_DSN\_NAME

If XRF\_DSN\_MEM > ' ' then, +

 DSNInfo = DSNInfo'('XRF\_DSN\_MEM')'

\*

 DSNInfo = DSNInfo '@stp:'XRF\_DSN\_JOBSTEP' ddn:'XRF\_DSN\_DDNAME

 entry = Translate(DSNInfo,'\_',' ')

 If Wordpos(entry,AlreadyDone) > 0 then $SkipRow = 'Y'

 AlreadyDone = AlreadyDone entry

*One Table Tool Model is for dataset names, and includes RELEATES for the DSN, job step and DDname*

J

//MODELDSN DD \*

 RELATE OBJECT

 '&DSNInfo'.

//MODELPGM DD \*

*Another Table Tool Model is for program name and the Load libraries where they are found*

 RELATE MEMBER &XRF\_DSN\_MEM

 DATASET = '&XRF\_PROC\_PROCLIB'

 INPUT.

//SYSEXEC DD DISP=SHR,DSN=CARSMINI.NDVR.R1801.CSIQCLS0

//SYSTSPRT DD SYSOUT=\* REXX MESSAGES

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

*RELATE statements are written here*

// UNIT=3390,SPACE=(TRK,(1,1)),

// DCB=(RECFM=FBA,LRECL=080,BLKSIZE=8000,DSORG=PS)

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

//\* Process Relates into Endevor \*

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

//RELATES2 EXEC PGM=CONRELE, < Endevor RELATES

Endevor’s CONRELE processor step, creates ACM relationships from the RELATE statements.

// COND=(1,NE,RELATES1)

//NDVRIPT DD DSN=&&RELATES,DISP=(OLD,DELETE)

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

//\* Store this listing.

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

//STORELST EXEC PGM=CONLIST,

Save the JCLCHECK listing into an Endevor listing library

// PARM='STORE',

// MAXRC=0,

// COND=EVEN

//C1BANNER DD UNIT=VIO,

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

// RECFM=FBA,LRECL=121

//C1LLIBO DD DSN=&LSTLIB,

// DISP=SHR,

// MONITOR=COMPONENTS

//LIST01 DD DSN=&&LIST01,DISP=(OLD,DELETE)

 Example outputs

RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(CLEANAP2) @stp:STEP6 ddn:TBLOUT'.

 RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(SYSTEMS) @stp:STEP7 ddn:TABLE'.

 RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(CLEANAP1) @stp:STEP7 ddn:OPTIONS1'.

 RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(CLEANAP2) @stp:STEP7 ddn:OPTIONS2'.

 RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(CLEANAPR) @stp:STEP7 ddn:TBLOUT'.

 RELATE OBJECT

 'PUBLIC.NDVR.OUTPUTS(CLEANAPR) @stp:SHOWME ddn:SYSUT1'.

 RELATE MEMBER IEBGENER

 DATASET = 'SYS1.LINKLIB'

 INPUT.

 RELATE MEMBER IRXJCL

 DATASET = '\*\*\*LPA\*RESIDENT\*\*\*'

 INPUT.

 RELATE MEMBER NDVRC1

 DATASET = 'CARSMINI.NDVR.R1801.CSIQAUTH'

 INPUT.

# 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