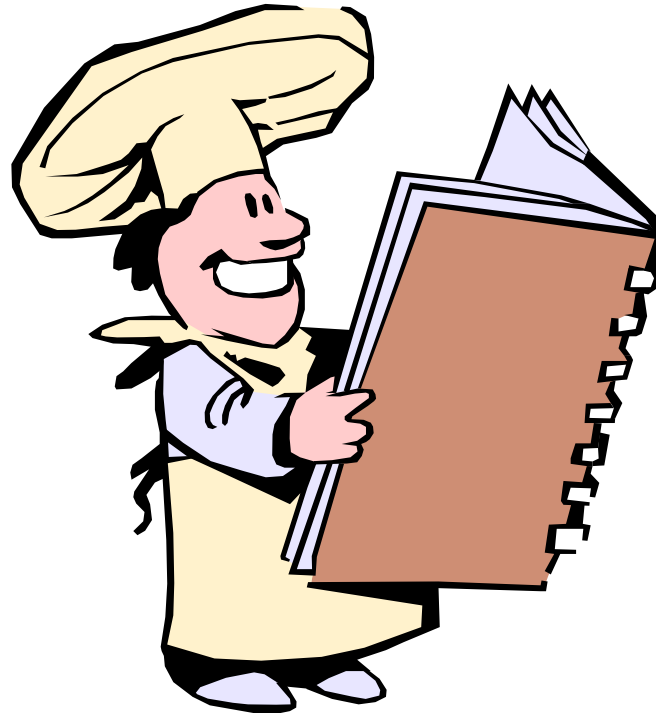


CA-Endevor

Quality Assurance for Job Control Language

EUNE – October 2007



Rose A. Sakach

Endevor Practice Leader - RSH Consulting, Inc.

R.Sakach@RSHConsulting.com - 617-969-9050 - www.rshconsulting.com

Abstract

CA-Endevor

Quality Assurance for Job Control Language

As the premier software change control tool for the mainframe, one aspect of CA-Endevor is to prepare software and its related configuration items and then stage them accordingly in the software life cycle.

This session will illustrate how to exploit Endevor's ability to improve quality, reduce risk, and eliminate inconsistencies in JCL by coding processors that will enforce quality assurance checking and automatically prevent forward migration of such configuration items. Topics include guidelines for interfacing Endevor with ASG's JCLPREP product which provides JCL validation and standards enforcement, coding samples for JCL validation, Endevor processor and processor group samples.

AGENDA

Job Control Language (JCL) Life Cycle

Interfacing with Endeavor

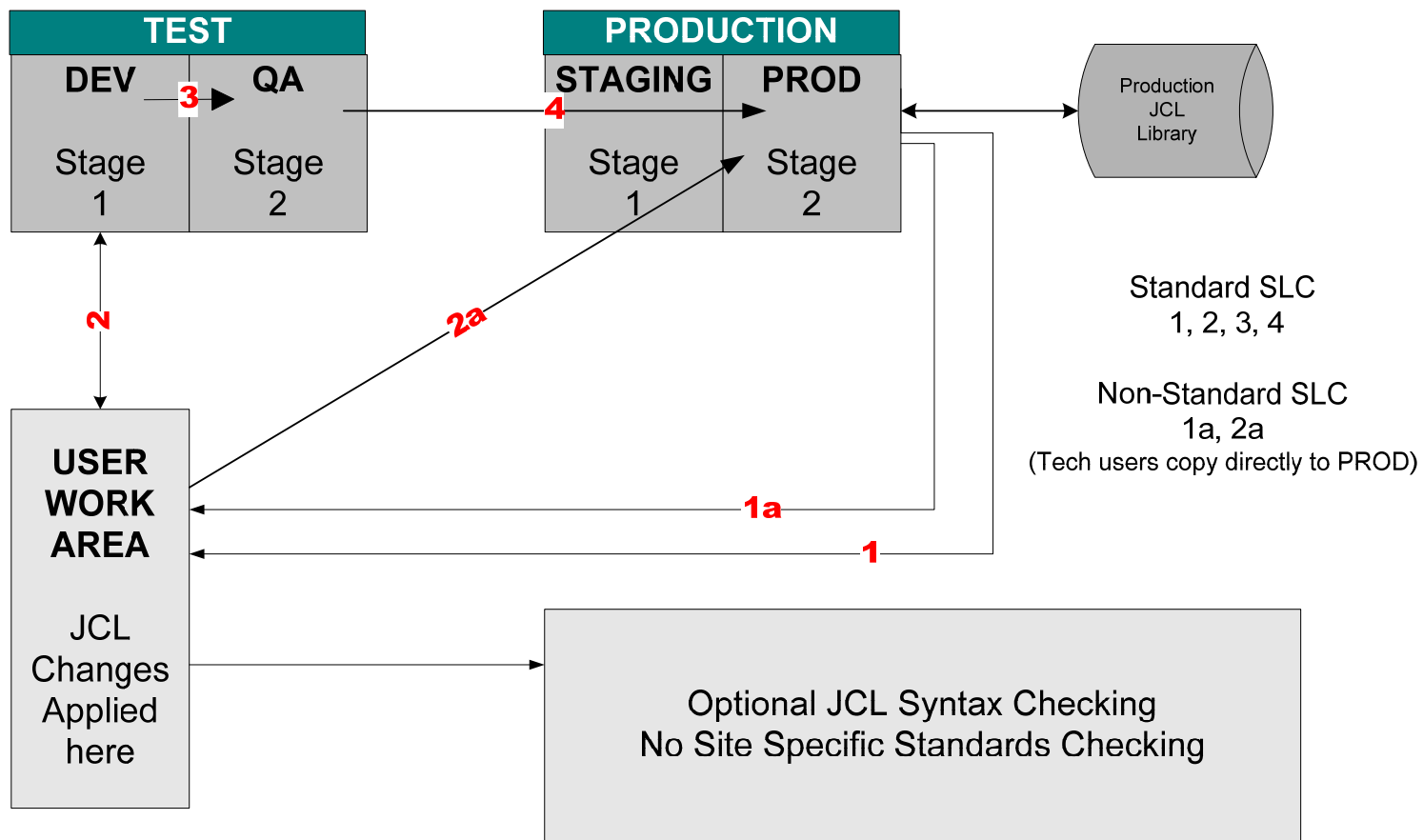
Customizing JCLPREP

User Training

Maintenance Recommendations

JCL Life Cycle

Current



JCL Life Cycle

Limited standardized, repeatable change control process

- Endeavor procedures for code migration are not enforced for all users
- JCL validation software use is optional



Inconsistent signoff / production implementation procedures (development vs. technical areas)

- Tech areas have more at risk and tend to use change control software the least – version control / recovery is a manual process
- Signoff does not require validation

JCL Life Cycle

Increased Security Exposures

- **Single, shared enterprise wide Library (typically a requirement of scheduling software) with many user ids requiring UPDATE access**
- **Unapproved use of surrogate ids in jobs**

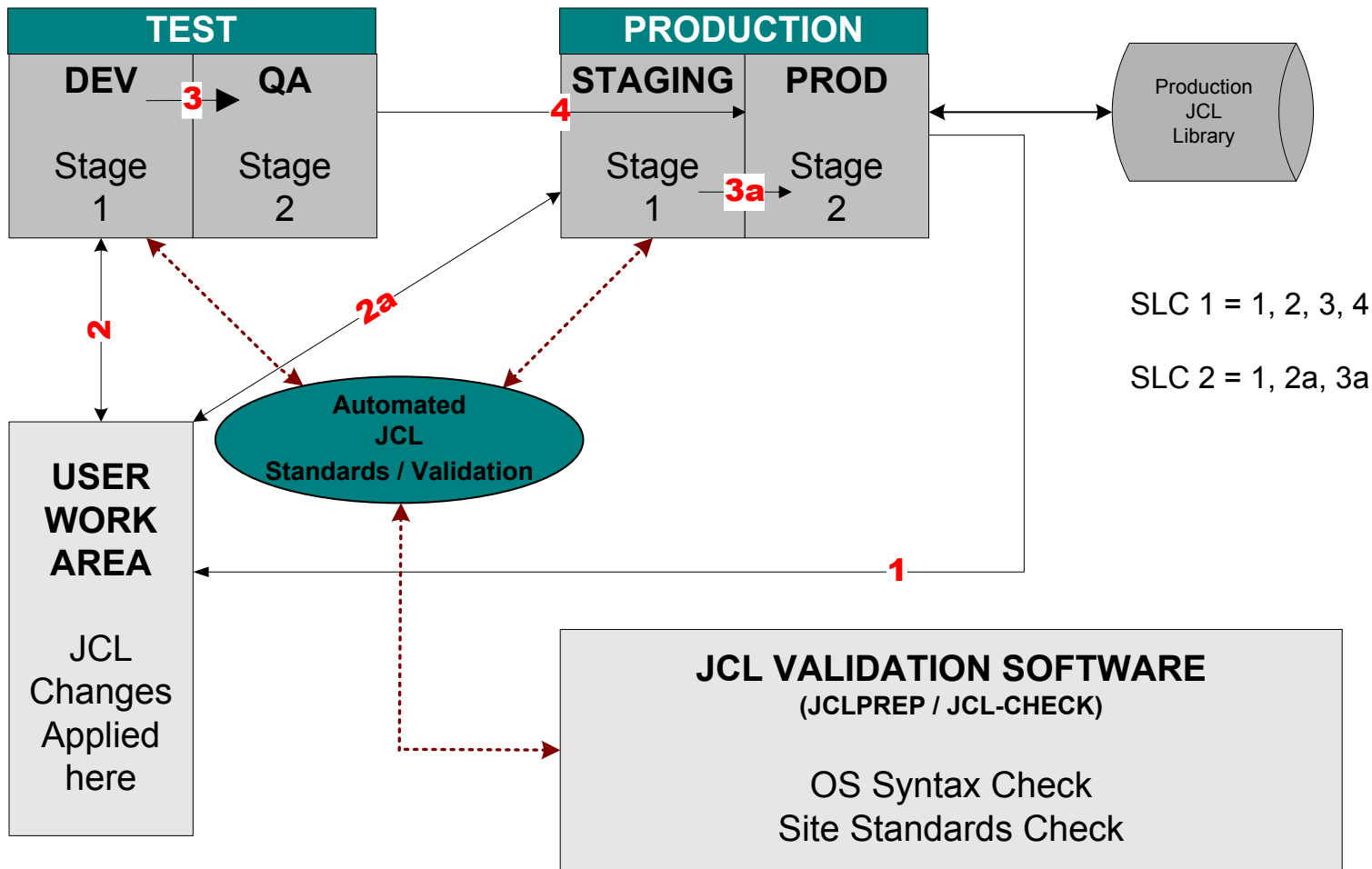


High risk of failure

- **Lack formal testing requirements**
 - **Test JCL isn't the same as Production JCL**
- **No automated validation**
- **Incorrect code can be migrated and propagated**
- **Changes can be regressed**

JCL Life Cycle

Proposed



Interfacing With Endeavor

Fully standardized, repeatable change control process

- Endeavor procedures for code migration **are** enforced for all users
- JCL validation software use is **not** optional



Consistent signoff / production implementation procedures (**for all areas**)

- Tech areas **follow the same** change control procedures – version control / recovery is **no longer** a manual process
- Signoff **includes** validation

Interfacing With Endeavor

Decreased Security Exposures

- **Only a single** user id (**Endeavor's alternate id**) requires UPDATE access to the enterprise JCL library (imposing a barrier between the users and the library)
- JCL validation software **validates and limits** use of surrogate ids



Greatly reduced risk of failure

- **Automated** validation
- **Non-Validated** code **cannot** be migrated and propagated
- Change regression is limited and can occur **only by ignoring warning messages**
- **Version control is automatic**
- **Recovery is fast**

Interfacing With Endeavor

JCL Update Process – Using Endeavor and JCL Software

Step 1

- User **RETRIEVES** a JCL member from Production inventory into or creates a new member in a work area PDS
- User modifies the JCL member accordingly

Step 2

- User **ADDS** the JCL member to Endeavor entry stage
 - Endeavor **GENERATE** processor automatically invokes the JCL syntax and JCL site standard checking tool and issues a non-zero return code if either check fails (Endeavor element is marked ***failed***)
- User reviews the **LISTING** library entry for any syntax or site standards coding errors that would cause the element to be marked ***FAILED***

Interfacing With Endeavor

JCL Update Process – Using Endeavor and JCL Software

Step 3

- **User MOVES the JCL element to the next stage in the life cycle for preparation of job scheduling considerations prior to Production implementation**
 - **Note: Element in *failed* status cannot be migrated to the next stage**
- **User resolves any outstanding JCL syntax errors**
- **User resolves any outstanding JCL site standards errors**

Step 4

- **User CREATES and CASTS a package to MOVE the JCL element to PRODUCTION**
- **User obtains package approvals**
 - **Note: Package approval may include a scheduling administrator. Emergency “Firecall” situations may use Emergency type packages / approvals.**
- **Package execution loads the JCL into the production library**

Interfacing With Endeavor

Specifications to Consider:

- **JCL validation required at entry stage**
- **JCL site specific standards validation required at entry stage**
- **Code separate rule for site standards**
- **Code alternative rules for technical staff (vendor code)**
- **Ensure site-specific symbolic code is translated appropriately prior to syntax check**
- **Provide a process and procedures for bypassing automated checks**
 - **Note: This should be a last resort and should not be permitted to propagate**
- **Provide a standardized process for reporting elements migrated via exception processing**
- **Thoroughly test JCL validation tool with all z/OS (and related software) upgrades**

Interfacing With Endeavor

Guidelines

- **Insert your JCL validation step in the GENERATE processor**
 - **Tip: Code (at least) two steps – 1 for JCL validation, 1 for site standards**
 - **Tip: Code a flag for each which will permit a bypass (as a failsafe)**
 - **Tip: Store any JCL “rules” (software) within Endeavor to take advantage of change tracking**
- **Utilize CONWRITE to capture and store the JCL in a temporary file**
- **Store validation listings (CONLIST) within Endeavor**
- **Ensure validation condition codes (issued from the JCL validation software) appropriately mark element add as *FAILED***

Interfacing With Endeavor

Entry Stage GENERATE Processor

... (Allocate temporary files)

... (Initialize listing libraries)

```
//*****  
//*STEP FUNCTION: EXTRACT THE JCL FROM ENDEVOR Source Mgmt LIBS  
//*****  
//CONWRITE EXEC PGM=CONWRITE, PARM='EXPINCL (&EXPINC)' ,MAXRC=0  
//ELMOUT DD DSN=&&SOURCE (&C1ELEMENT) ,  
// UNIT=SYSDA ,  
// SPACE=(TRK, (p, s, d) ,RLSE) ,  
// DISP=(NEW, PASS, DELETE) ,  
// DCB=(LRECL=80, RECFM=FB, BLKSIZE=0)
```

(Continued on next slide)

Interfacing With Endeavor

Entry Stage GENERATE Processor (Site translation)

(Continued from previous slide)

```

//*****
//*   STEP FUNCTION: Invoke JCL Check for site translation
//*****
//JCLTRAN EXEC PGM=JCLPREP,COND=(0,LT,CONWRITE),
//          MAXRC=4,EXECIF=(&JCLTRANS,EQ,'Y')
//SYSUDUMP DD SYSOUT=*
//DDIN     DD DSN=&&SOURCE,DISP=(OLD,PASS)
//DDOUT    DD DSN=&&JCLSRCE,DISP=(OLD,PASS)
//DDXEFI   DD DSN=&RULESLIB(&RSPJTRN),DISP=SHR
//DDOPT    DD DSN=&JOPTSLIB(&OSPJTRN),DISP=SHR
//DDRUN    DD *
PDS
INCLUDE &C1ELEMENT
/*
//DDWORK1  DD DSN=&&WORK01,
//          DISP=(OLD,PASS)
//DDWORK2  DD DSN=&&WORK02,
//          DISP=(OLD,PASS)
//DDRPT    DD &&SYNLIST,DISP=(OLD,PASS)
//DDXEFP   DD DUMMY
//DDXEFW   DD DUMMY

```

(Continued on next slide)

Interfacing With Endeavor

Entry Stage GENERATE Processor (Syntax Check)

(Continued from previous slide)

```

//*****
//*   STEP FUNCTION: Invoke JCL Check for SYNTAX validation
//*****
//JCLSYN  EXEC PGM=JCLPREP,COND=(4,LT,JCLTRAN) ,
//        MAXRC=4, EXECIF=(&SYNCHECK,EQ,'Y')
//SYSUDUMP DD SYSOUT=*
//DDIN    DD DSN=&&JCLSRCE,DISP=(OLD,PASS)
//DDOUT   DD DUMMY
//DDXEFI  DD DSN=&RULESLIB(&RSPJSYN),DISP=SHR
//DDOPT   DD DSN=&JOPTSLIB(&OSPJSYN),DISP=SHR
//DDRUN   DD *
PDS
INCLUDE &C1ELEMENT
/*
//DDWORK1 DD DSN=&&WORK01,
//        DISP=(OLD,PASS)
//DDWORK2 DD DSN=&&WORK02,
//        DISP=(OLD,PASS)
//DDRPT   DD &&SYNLIST,DISP=(OLD,PASS)
//DDXEFP  DD DUMMY
//DDXEFW  DD DUMMY

```

(Continued on next slide)

Interfacing With Endeavor

Entry Stage GENERATE Processor (Site Standards Check)

(Continued from previous slide)

```

//*****
//*   STEP FUNCTION: Invoke JCL Check for STANDARDS validation
//*****
//JCLSTD  EXEC PGM=JCLPREP,COND=(4,LT,JCLTRAN) ,
//          MAXRC=4, EXECIF=(&STDCHECK,EQ,'Y')
//SYSUDUMP DD SYSOUT=*
//DDIN     DD DSN=&&JCLSRCE,DISP=(OLD,DELETE)
//DDOUT    DD DUMMY
//DDXEFI   DD DSN=&RULESLIB(&RSPJSTD),DISP=SHR
//DDOPT    DD DSN=&JOPTSLIB(&OSPJSTD),DISP=SHR
//DDRUN    DD *
PDS
INCLUDE  &C1ELEMENT
/*
//DDWORK1  DD DSN=&&WORK01,
//          DISP=(OLD,PASS)
//DDWORK2  DD DSN=&&WORK02,
//          DISP=(OLD,PASS)
//DDRPT    DD &&SYNLIST,DISP=(OLD,PASS)
//DDXEFP   DD DUMMY
//DDXEFW   DD DUMMY

```

(Continued on next slide)

Interfacing With Endeavor

Entry Stage GENERATE Processor

(Continued from previous slide)

... (BSTCopy to Output Library)

... (Store Listings)

... (Print Listings)

Interfacing With Endeavor

Bypassing JCL Checks

PROCESSOR GROUP = **JCL**

			BACKGROUND EXECUTION	FOREGROUND EXECUTION
GENERATE PROCESSOR:	GJCL0000	:	Y (Y/N)	Y (Y/N)
DELETE PROCESSOR:	*NOPROC*	:	Y (Y/N)	Y (Y/N)
MOVE PROCESSOR:	*NOPROC*	:	N (Y/N)	N (Y/N)

PROCESSOR GROUP = **JCLNOCK1**

			BACKGROUND EXECUTION	FOREGROUND EXECUTION
GENERATE PROCESSOR:	GJCL0000	:	Y (Y/N)	Y (Y/N)
DELETE PROCESSOR:	*NOPROC*	:	Y (Y/N)	Y (Y/N)
MOVE PROCESSOR:	*NOPROC*	:	N (Y/N)	N (Y/N)

Notes:

Standard Processor Group: SYNCHECK=Y; STDCHECK=Y

Bypass Processor Group: SYNCHECK=Y; STDCHECK=**N**

Customizing JCLPREP

JCLPREP Basics

- Executables called “rules” are invoked to perform syntax, site standards validation, and any “standalone” JCL manipulation
- The rules are coded in a 4GL language XEF (eXpert Editing Facility) Language
- Rules are like programs
 - R@ prefixed members are “unprepared” or non-compiled rules and can be standalone type routines or can be included in other R@ members to produce a single P\$ rule or executable
 - R@UEXIT is the standard user exit routine provided by the vendor - this routine can be executed as is, or can be customized to include site specific rules
 - P\$ (or U\$) prefixed members are “prepared” or compiled rules

Customizing JCLPREP

Suggested Naming Conventions

R@xxxxxx

- Vendor provided source code (do not modify)
- Where xxxxxx = 6 character description of the routine (i.e. R@ENTER, R@DEFJCL, R@INIT, R@SETUP, R@UEXIT)

U@UXITxx

- User modified USER EXIT include
- Where xx = 2 character code

Customizing JCLPREP

Suggested Naming Conventions

U@xxxxxx

- User created source code include
- Where xxxxxx = 1- 6 character description of the routine (i.e. DD, JLIB, JOB, PROC etc.)

U#xxxxxx

- User created subroutine (coded within an “@” member via a LABEL) - can be CALLED from other included members
- Where xxxxxx = 1- 6 character description of the routine (i.e. DD, JLIB, JOB, PROC etc.)

Customizing JCLPREP

Suggested Naming Conventions

P\$xxxxxx

- Vendor provided executable rule (do not modify)
- Where xxxxxx = 6 character description of the routine (i.e. P\$RULE1)

U\$xxxxxx

- User created and prepared rule
- Where xxxxxx = 1- 6 character description of the routine (i.e. JCLRPT or STDCK)

Customizing JCLPREP

Customizing Rules using User Exits

- Document site standards
- Make a copy of R@UEXIT (i.e. save it as member U@UXIT00)
- Locate EXIT points and prepare to insert new routines
- Code U@xxxxxx (i.e. U@DD0s, U@DD0sa) subroutines to process the standards checks
- Create the new rule (replace R@UEXIT w/ U@UXIT00 in the source code for the basic rule set)
- Code meaningful error messages
- Prepare the new rule (U\$STD00)
- Make new rule available by adding it to the rules selection list (JCLPREP panels)
- Test the rule (interactive and batch)
- Add the rule to the appropriate Endeavor Processor(s)

Customizing JCLPREP

Sample User Exit

*XEFR0V6

*Unprepared Rule: U@UXIT00

*Prepared Rule: U\$STD00

*Description: Syntax / Site Standards Rule for Endeavor JCL types

*Site: Newton

*Notes: Calls In-house routines @ exit points

*Modification History

*Date	Who	What
-------	-----	------

*----	---	-----
-------	-----	-------

*4.14.04	RAS	Add Standards Checks
----------	-----	----------------------

INCLUDE R@ENTER

INCLUDE R@DEFJCL

INCLUDE R@DEFUSR

INCLUDE R@INIT

.

.

.

INCLUDE R@SETUP

...

(continued on next slide)

Customizing JCLPREP

Sample User Exit (continued from previous slide)

UEXITJOB:

RETURN 0

UEXITOPTS:

RETURN 0

UEXITDD:

RETURN 0

UEXITPROC:

RETURN 0

... (UEXITPEND, UEXITJOB1)
(continued on next slide)

Customizing JCLPREP

Sample User Exit (continued from previous slide)

```
*****
*User exit for DD card processing within a proc. This exit is
*called after symbolic substitution has been performed.
*Entry Conditions:
*      Parm1      -- The Statement number of the statement.
*      Parm2      -- The Proc step name.
*      Parm3      -- The Exec step name.
*      Parm4      -- The membername (if a PDS member).
*      Parm5      -- Name Field of the card.
*      Parm6      -- The scope number of the statement.
*      Parm7      -- Nesting level number.
*Note: This exit should be used for JCL examination since all
*      symbolics are resolved. Use exit UEXITDD if changes
*      must be made to the JCL.
*****
UEXITDD1:
*
* RSH Custom Rules
*
  INCLUDE U@DD0S
  INCLUDE U@DD0S1
*
RETURN 0
```

Customizing JCLPREP

Site Standard Validation Checks to Consider

- **Jobcard**
 - **Jobname**
 - **Account Code**
 - **CLASS**
 - **MSGCLASS**
 - **TIME**
 - **REGION**
 - **USER**

- **JCLLIB Order Statement**

- **JOBLIB / STEPLIB**

Customizing JCLPREP

Site Standard Validation Checks to Consider

- **DD statements**
 - **Application HLQ restrictions; no USERID or TEST datasets allowed etc.**
 - **Unit type / Jobclass limitations (i.e. tapes)**
 - **Dataset dispositions**
 - ◆ **Space coding restrictions**
 - ◆ **SMS dataset name qualifiers**
 - ◆ **DCB (BLKSIZE=0 etc.)**
 - **Tape specifications**
 - ◆ **EXPDT**
 - ◆ **GDG Pattern DSCB**
 - ◆ **RETPD**
 - **SYSOUT**

- **Abend or user-to-be-contacted instructions (embedded comments)**

Customizing JCLPREP

Sample XEF custom rule - 1

U@DDOS:

```
*****
*User Include:      U@DDOS (Label = n/a)
*Called by:        -----
*Included by:      U@UXIT00
*Description:      DD Statement Site Standard Checks
*Site:             Newton
*Notes:           JCL, SPECJOB, SPECJOBH types
*****
*User Variables - Integers; Alphanumeric; Tables
  DEFUVAR DSNlen   INT
  DEFUVAR FLQlen   INT
  DEFUVAR FQChar   Len(1)
  DEFUVAR TLQlen   INT

*For JOBLIB/STEPLIB table
  DEFUVAR LibIndex      INT
  DEFUVAR LibArray(30)  Len(44)
  DEFUVAR Jlibfile      Len(18)
  DEFUVAR JSlib         Len(8)
  ...
(continued on next slide)
```

Customizing JCLPREP

Sample XEF custom rule - 1 (Continued from previous slide)

```
*User Variables - Inits
  Init LibIndex = 0
  ...
*
*Check for NULL dsn; break DSN by nodes
*
  If DSN EQ Null Then
    GoTo DD0SExit
  End

  Call Dsnnode(DSN)
*
*Check JOBLIB / STEPLIB
*
  Set JSlib = STMTLABEL
  If ((JSlib EQ "JOBLIB") or (JSlib EQ "STEPLIB")) Then
    Call U#JBLIB
    GoTo DD0SExit
  End
  .....
```

(continued on next slide)

Customizing JCLPREP

Sample XEF custom rule - 1 (Continued from previous slide)

...

CkNewDSN:

...

...

CK2Exit:

Return

CkTape:

...

...

CkTapeExit:

Return

DD0SExit:

Return 0

Customizing JCLPREP

Sample XEF custom rule - 1a

U@DDOS1:

```
*****
*User Include:      U@DDOS1 (Label = U#JBLIB)
*Called by:        U@DDOS
*Included by:      R@EXIT0S
*Description:      Validates JOBLIB / STEPLIB libraries against
*                  A list stored in DSN:  OPS.NDVR.PREPJLIB
*Site:             Newton
*Notes:            JCL, SPECJOB, SPECJOBH types
*****
*
*COMessge defined in U@JCARD1 as:
*      DEFUVAR CoMessge Len(20)
*Init CoMessge = "RSH Site Standard:  "
*
*User variables - Defined in U@DDOS
*****
* Read JOBLIB / STEPLIB file to validate
*****
*Note: Endeavor libraries are valid JOBLIB / STEPLIBs
```

(continued on next slide)

Customizing JCLPREP

Sample XEF custom rule - 1a (Continued from previous slide)

U#JBLIB:

```
  If Node1 EQ "ENDV" then
    GoTo DD0S1XIT

  End

*Store valid entries in array
*
  Set Jlibfile = "OPS.NDVR.PREPJLIB"
  Call DSNSOURCE("Libarray","libIndex",Jlibfile)
  If (RC NE 0) then
    Report "*UADD0S1 Error -- Reading File" .. Jlibfile ..,
    "RC=" .. Rc
    GoTo DD0S1XIT
  End
```

(continued on next slide)

Customizing JCLPREP

Sample XEF custom rule - 1a (Continued from previous slide)

```
*  
*Scan Array  
*  
  Set Index1 = 1  
  Set DSNlen = Length(DSN)  
  Call DUP("Libarray",DSN,Index1,DSNlen,"index2")  
  
  If (Index2 EQ 0) Then  
    errnum = 0332  
    errsev = 08  
    rptbuf = CoMessge || "Invalid JOBLIB / STEPLIB"  
    call errcommon  
  
  End  
*****  
DDOS1XIT:  
*****  
Return 0
```

User Training

Create Procedures Document and make it accessible

Create FAQ Document

- **Sample error messages**
- **Where to find the listing**
- **What to do when an element will not MOVE to the next stage**

Provide Messages Guidelines

Custom Error Messages Code Range:

- **0800 – 0899 Site Translation (ALL JCL Types)**
- **0900 – 0999 TEST Standards (TESTJCL, TESTPROC)**
- **1000 – 1999 PROD Standards (JCL, PROC)**
- **9000 – 9999 Stand Alone**

User Training

Notice of Automated JCLPREP / Endeavor Interface

Effective Date: 4/22/07

Endeavor Systems: PAY

The Endeavor Systems listed above will be modified to include an automated JCLPREP check when the following element types are added to the first stage of any environment: 1. **JCL** 2. **PROC** 3. **TESTJCL** 4. **TESTPROC**

Questions? Concerns? Please send an E-Mail to: ENDEVOR SUPPORT

If the JCLPREP check completes with a Return Code ≥ 08 , the add to Endeavor will be unsuccessful and the element will be marked in the MCF with a status of ***FAILED***.

Note: You will be responsible for resolving all JCL errors prior to migrating the element to Production.

NOTE: The Endeavor MOVE function will FAIL if the current element status = ***failed***.

Maintenance Recommendations

Provide method for reporting “issues” (i.e. errors that should be warnings etc.)

Communicate with z/OS systems programmer

Create reusable test plan for OS upgrades and JCLPREP upgrades

Report on any JCL members using the “bypass” processor groups regularly and follow-up

Summary

Interfacing Endeavor with a JCL Validation Tool Provides:

- A standardized, repeatable process which automatically enforces customized site standards and JCL validation while preventing syntax errors from migrating forward in the software life cycle
- Version control, exception processing, and audit trails which assist in preventing code regression, provide the ability to perform timely back-outs, and allow for monitoring, reporting, and follow-up activities.

Configuration involves:

- Implementing customized JCL rules via user exits to validate site standards
- Coding Endeavor JCL generate processor which invokes the JCL tool to perform JCL syntax checking and the customized site standards
- Defining processor groups which permit rule bypass according to company policy
- Documenting and implementing follow-up procedures for exceptions

Questions

