



Using z/OS 2.2 JES3 New Function

David Jones
IBM JES3 Development



August 13, 2015
Session 17600 at 8:30 – 9:30 AM



#SHAREorg



SHARE is an independent volunteer-run information technology association
that provides **education, professional networking and industry influence.**

Copyright (c) 2015 by SHARE Inc. Except where otherwise noted, this work is licensed under
<http://creativecommons.org/licenses/by-nc-sa/3.0/>



Trademarks



The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

IBM®
MVS
JES2
JES3
RACF®
z/OS®
zSeries®

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Complete your session evaluations online at www.SHARE.org/Orlando-Eval

8/13/2015



What's to be covered

- More details on JES3 V2R2 support of:
 - Symbols
 - JCL support
 - Tracking JECL usage
 - JES3 scaling
 - JES3 health checks

JES3 symbols support



- V2R1 JES3 supported system symbol substitution for BATCH jobs.
 - BATCH JCL can refer to any symbol in IEASYMxx member.
 - **SYSSYM=DISALLOW|ALLOW** option added for a job CLASS.
 - Symbol substitution occurs during conversion of the JCL.
- V2R2 JES3 extends the symbols support.
 - JCL symbols can be made available to the running job.
 - JES symbols can be created dynamically.
 - JCL and JES symbols can be passed through INTRDR to a submitted job.
 - Symbol substitution in instream data sets.

JES3 symbols support

Three types of symbols



- **System symbols**
 - Specific to the MVS system.
 - Defined in in the IEASYMxx member of SYS1.PARMLIB.
- **JCL symbols**
 - JCL Symbols are defined within the JCL.
 - By default, available to the job only during conversion.
 - Can be made available to the job during run time.
- **JES symbols**
 - Dynamic symbols at the job step level or current task level.
 - Created and managed by the JES Symbol Service.

JES3 symbols support

Export JCL symbols



- Previously, JCL symbols were used only during JCL conversion when the JCL stream of a job was processed.
 - Symbols were not saved as part of the converter output.
- **EXPORT** JCL statement places the symbols in converter output for use by the job during run time.
 - Specify the symbols to be made available.
 - Next setting of the symbol will be exported.
 - Scope of any symbol is the step.
- The exported symbols can be used in a number of ways:
 - Programmatically accessed using the JCL Symbol Service (IEFSJSYM) or the JES Symbol Service (IAZSYMBL).
 - Passed to jobs submitted through the internal reader.
 - Used for substitution within instream data.

JES3 symbols support EXPORT JCL statement



- **EXPORT** JCL statement syntax:

```
//MYEXPRT  EXPORT  SYMLIST=(sym1 , sym2 , ...)
```

 - Symbols identified with parameter **SYMLIST**=(sym1,sym2,...)
 - List of JCL symbol names without the '&' character
 - JCL symbol syntax rules apply
 - SYMLIST=* exports all JCL symbols
- **EXPORT** can appear anywhere after the **JOB** statement.
 - Only need to identify an exported symbol once in the JCL.
 - Export applies to all steps after the **EXPORT** statement.
 - **SET** JCL statement required after **EXPORT** statement to export a value.
 - Last value **SET** for a step is the value exported.
 - Only one value of a JCL symbol exported per job step.

JES3 symbols support

EXPORT example 1



```
//EX1   EXPORT  SYMLIST=(S1,L1)
//SET1   SET     S1=STEWART,J1=JFK,N1=NIAGARA,L1=LAX
//SET2   SET     S1=SANDIEGO,F1=FRESNO
//EX2   EXPORT  SYMLIST=F1
//STEP1 EXEC   PGM=USERPGM1
//STEP2 EXEC   PGM=USERPGM2
//SET3   SET     S1=MSP
```

- Symbols exported for STEP1 (available at run time) are:
 - S1 with value SANDIEGO
 - L1 with value LAX
 - F1 with null value (not SET after EXPORT)
- Symbols exported for STEP2 (available at run time) are:
 - S1 with value MSP (SET in STEP2)
 - L1 with value LAX
 - F1 with null value (not SET after EXPORT)

JES3 symbols support

EXPORT example 2



```
//EX1      EXPORT  SYMLIST= (DSN, MEMB)
//SET1      SET     DSN= 'SYS1.SAMPLIB '
//SET2      SET     MEMB=SAMP2
//STEP1     EXEC    PGM=USERPGM1
//INPUT     DD      DSN=&DSN (&MEMB)
//SET3      SET     MEMB=SAMP3
//OUTPUT    DD      DSN=&DSN (&MEMB)
```

- Symbols exported for STEP1 (available at run time) are:
 - DSN with value SYS1.SAMPLIB
 - MEMB with value SAMP3
- Note that MEMB had 2 values SET in the JCL for STEP1.
 - SAMP2 from SET2 and SAMP3 from SET3
 - Only one value of a JCL symbol exported per job step.
 - Last SET statement value is the value exported.

JES3 symbols support JCL symbol service



- JCL symbol service **IEFSJSYM** gives read-only access exported JCL symbols at run time.
 - Symbols are static and cannot be modified.
 - Symbol names, length, and value follow JCL rules.
 - Scope of the symbols is a job step including all tasks running in the job step.
 - Available after step has started running.
 - i.e. Not available in exits like IEFUJI
 - Also available via LE interface CEEGTJS

JES3 symbols support JCL symbol service



- IEFSJSYM usage:
 - Two request types:
 - REQUEST=GETALL returns all exported JCL symbols and values for the job step.
 - REQUEST=GETBYNAME returns values for the JCL symbol names provided.
 - Generic names allowed using '*' and '?' wildcards.
 - Output storage area provided by caller mapped by IEFSJSYD.
 - Symbols names do not include '&' character.

JES3 symbols support

JES symbols



- New type of symbol introduced with V2R1...
 - ...can have names up to 16 characters long and values up to 4K bytes.
 - ...can be created/updated/deleted during run time.
 - Created at the job step or task level.
 - ...can be used for in-stream symbol substitution.
 - ...can be passed on to a submitted job via INTRDR.
 - ...can be used internally by an application and to communicate between programs within the same job step.
 - ...can be used to communicate information between applications and JES with special purpose symbols.
 - ...do not have to be explicitly exported.

JES3 symbols support

JES symbol service



- JES symbol service **IAZSYMBL** manages JES symbols.
 - Creates symbols at the job step or task level.
 - A JES symbol overrides a JCL symbol of the same name.
 - A JES symbol at the task level overrides a job level JES symbol of the same name.
 - Can be used to read exported JCL symbols.
- **IAZSYMDF** data definition macro
 - Maps parameter structure passed to IAZSYMBL.
 - Filters, options, return and reason codes, feedback data.
 - Maps data structures passed to or returned from IAZSYMBL.
 - Data to create and update JES symbols.
 - Extracted symbols data.

JES3 symbols support

JES symbol service



- IASYMBL usage:
 - Operations:
 - CREATE – create one or more JES symbols and set initial values.
 - UPDATE – update values of specified JES symbols.
 - CLEAR – clear values of specified JES symbols.
 - EXTRACT – retrieve values of specified JES and JCL symbols.
 - DELETE – delete the specified JES symbols.
 - Can operate on multiple JES symbols with one call including using ‘*’ and ‘?’ wildcards for EXTRACT and DELETE operations.
 - EXTRACT operation searches for requested symbols in the order:
 - Task level JES symbol
 - Job level JES symbol
 - Exported JCL symbol

JES3 symbols support

Passing symbols through INTRDR



- JES and JCL symbols can be passed from a parent job to a submitted job on the INTRDR.
 - Parent job and submitted job can use a consistent set of symbols.
 - Symbols are treated as having been SET after the JOB statement.
 - Symbols are available for substitution during JCL processing.
 - Same as symbols which have been explicitly SET.
 - Implicitly exported within in the submitted job for run time use.
 - EXPORT statement not required for passed symbols.
- Symbols which can be passed:
 - JCL symbols available to a parent job.
 - JES symbols that conform to JCL requirements for symbol name and value length.
 - e.g. cannot pass a JES symbol with a 4K value.

JES3 symbols support

Passing symbols through INTRDR



- **INTRDR** allocation syntax:

```
//SYSUT2 DD SYSOUT=(A,INTRDR) ,  
//          SYMLIST=(sym1 , sym2 , ...)
```

- Symbols identified with parameter **SYMLIST**=(sym1,sym2,...)
 - List of JCL or JES symbol names without the '&' character
 - JCL symbol syntax rules apply
 - SYMLIST=* exports all JCL compatible symbols
- Symbol values can be changed between job submits.
- Values are extracted when INTRDR is closed or with ENDREQ.
 - JES2 extracts when the JOB card is processed by INTRDR.
- Dynamically allocated internal reader has equivalent function
 - Text unit DALSYML (TU key X'802B')

JES3 symbols support INTRDR feedback



- Internal reader sets a symbol for jobs processed.
- `SYS_LASTJOBID`
 - Set when job submitted through internal reader is successfully accepted by JES (job submission succeeded).
 - The value of this symbol is 8-character JES job identifier of the job which was just submitted.
 - Set to null value when job submission failed.
- Use JES Symbol Service (IAZSYMBL) to EXTRACT the value.
 - `SYS_LASTJOBID` is a task level symbol.
 - Extract must be done in same task as the job submit.

JES3 symbols support

Instream symbol substitution



- Symbols can now be used in instream data (similar to JCL)
 - Instream data created by JES or converter
- Symbol substitution occurs when an application reads an instream data set record.
 - Using current value of the symbol.
- Symbols can come from:
 - JCL symbols made available using the EXPORT JCL statement.
 - JES symbols created using JES Symbol Service (IAZSYMBL).
 - MVS system symbols from either conversion or execution system.
- Optional data set can be used for logging the results.
 - Logs original and substituted text.
- Substitution controlled by SYMBOLS parameter on DD JCL statement.

JES3 symbols support

Instream symbol substitution

- **DD DATA** or **DD *** JCL statement parameter **SYMBOLS=** to request JES to perform instream symbol substitution.
 - **SYMBOLS=**([JCLONLY|EXECSYS|CNVTSYS[,*DDname*])
 - **JCLONLY** – Substitute exported JCL symbols and JES Symbols
 - **EXECSYS** – JCLONLY and system symbols defined on the system during job execution.
 - **CNVTSYS** – JCLONLY and system symbols from the system where the job has undergone JCL conversion.
 - ***DDname*** – DD name for a log data set provided by the user.
 - **SYMBOLS=** not specified means no substitution occurs!

JES3 symbols support

Instream symbol substitution rules

- Symbols and value can be different lengths.
 - Substitution attempts to preserve columns by compressing out blanks to right of a symbol.

Given: &SYSNAME=SY1 &DSN=ABC.DATA.TEXT &VOL=WORK12

12345678901234567890123456789012345678901234567890123456789012...

Source: SYSTEM=&SYSNAME, DSN=&DSN, VOLUME=&VOL Comment

Result: SYSTEM=SY1, DSN=ABC.DATA.TEXT, VOLUME=WORK12 Comment

- If not enough blanks, data is shifted right.

Given: &SYSNAME=SY1 &DSN=ABCDEFG.DATASET.TEXT &VOL=WORK12

12345678901234567890123456789012345678901234567890123456789012...

Source: SYSTEM=&SYSNAME, DSN=&DSN, VOLUME=&VOL Comment

Result: SYSTEM=SY1, DSN=ABCDEFG.DATASET.TEXT, VOLUME=WORK12 Comment

- Data can be shifted beyond LRECL.
 - Will present long record to application (potentially causing I/O error).
 - Application can attempt to recover from error by passing longer buffer.
- JCL coder's responsibility to ensure enough room for substitution

JES3 symbols support

Instream symbol substitution example



- Job segment using system, JCL, and JES symbols.

```
//          EXPORT SYMLIST=(DSN,VOL)
//          SET     DSN='ABC.DATA',VOL='123456'
//STEP1     EXEC    PGM=USERPGM1
//DATA      DD     DSN=&DSN,DISP=SHR
//SYSIN     DD     *,SYMBOLS=EXEC SYS
              SYSTEM=&SYSNAME,DSNAME=&DSN,VOLUME=&VOL
              FUNCTION=' &APPL_NAME '
/*
```

- Application reading SYSIN data on SY1 will see the records as:

```
SYSTEM=SY1,DSNAME=ABC.DATA,VOLUME=123456
FUNCTION='RECORD SEARCH'
```

JES3 symbols support

Instream symbol substitution logging example



- Job segment using system, JCL, and JES symbols.
 - Application USERPGM1 creates JES symbol APPL_NAME with value 'RECORD SEARCH'

```
//          EXPORT SYMLIST=(DSN,VOL)
//          SET      DSN='ABC.DATA',VOL='123456'
//STEP1     EXEC     PGM=USERPGM1
//DATA      DD      DSN=&DSN,DISP=SHR
//MYLOG     DD      SYSOUT=A
//SYSIN     DD      *,SYMBOLS=(EXEC SYS,MYLOG)
              SYSTEM=&SYSNAME,DSNAME=&DSN,VOLUME=&VOL
/*
```

- MYLOG data set will show the substitution results:

```
SYSIN : RECORD 1 BEFORE SUBSTITUTION
SYSIN : SYSTEM=&SYSNAME,DSNAME=&DSN,VOLUME=&VOL
SYSIN : RECORD 1 AFTER SUBSTITUTION
SYSIN : SYSTEM=SY1,DSNAME=ABC.DATA,VOLUME=123456
```

JES3 symbols support

INTRDR example



- Job SYMSAMP submitting job CATALOG with INTRDR symbols

```
//SYMSAMP  JOB MSGLEVEL=(1,1),MSGCLASS=A,NOTIFY=IBMUSER
//          EXPORT SYMLIST=(DSN,VOLSER)
//          SET     DSN=TEST.JES.LINKLIB,VOLSER=STORAG
//STEP1     EXEC PGM=IEBGENER
//SYSPRINT  DD SYSOUT=*
//SYSUT2    DD SYSOUT=(A,INTRDR),SYMLIST=*
//SYSIN     DD DUMMY
//SYSUT1    DD DATA,DLM='%%'
//CATALOG   JOB 1,CATALOG,MSGLEVEL=(1,1),CLASS=A
//CATUSER   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=A
//SYSIN     DD *,SYMBOLS=(JCLONLY,SYMLOG)
//  DEFINE  NONVSAM (NAME(&DSN) DEVT(3390) VOL(&VOLSER)) -
//          CAT(PAGE08.CATALOG)
//SYMLOG    DD SYSOUT=A
%%
```

JES3 symbols support

INTRDR example - results

- Job CATALOG from JESJCL data set:

```
//CATALOG JOB 1,CATALOG,MSGLEVEL=(1,1),CLASS=A
// SET DSN=TEST.JES.LINKLIB
//DSN      EXPORT EXPSET=TEST.JES.LINKLIB
// SET VOLSER=STORAG
//VOLSER   EXPORT EXPSET=STORAG
//CATUSER  EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN    DD *,SYMBOLS=(JCLONLY,SYMLOG)
//SYMLOG   DD SYSOUT=A
```

GENERATED STATEMENT
GENERATED STATEMENT
GENERATED STATEMENT
GENERATED STATEMENT

- Generated statements result in the implicit SET and EXPORT for passed symbols.

JES3 symbols support

INTRDR example - results



- IDCAMS SYSPRINT data set shows substitutions:

IDCAMS SYSTEM SERVICES

```
DEFINE NONVSAM (NAME(TEST.JES.LINKLIB) DEVT(3390) VOL(STORAG)) -  
CAT(PAGE08.CATALOG)
```

IDC0001I FUNCTION COMPLETED, HIGHEST CONDITION CODE WAS 0

IDC0002I IDCAMS PROCESSING COMPLETE. MAXIMUM CONDITION CODE WAS 0

- SYMLOG data set showing JCL before and after:

SYSIN : RECORD 1 BEFORE SUBSTITUTION

```
SYSIN : DEFINE NONVSAM (NAME(&DSN) DEVT(3390) VOL(&VOLSER)) -
```

SYSIN : RECORD 1 AFTER SUBSTITUTION

```
SYSIN : DEFINE NONVSAM (NAME(TEST.JES.LINKLIB) DEVT(3390) VOL(STORAG)) -
```

SYSIN : RECORD 2 BEFORE SUBSTITUTION

```
SYSIN : CAT(PAGE08.CATALOG)
```

SYSIN : RECORD 2 AFTER SUBSTITUTION

```
SYSIN : CAT(PAGE08.CATALOG)
```

Job Notification ENF

- ENF 78 for notification of job completion.
 - Requested by applications creating JES symbol SYS_JOB_NOTIFY (use JES symbol service)
 - Value can be set with up to 4K of data (JES symbol).
 - Create prior to submitting JCL through the internal reader.
- ENF 78 issued when job moves beyond execution phase.
 - Includes job identification and completion information.
 - Includes the value of the SYS_JOB_NOTIFY symbol.

JES3 JCL support

// JCLLIB PROCLIB=



- **PROCLIB=** specifies PROCLIB concatenation for a job.
 - Overrides the procedure library specified with PROC=xx keyword on the //*MAIN JECL statement.
 - Message IAT6157 written to JESMSG LG data set indicating the override of PROC=xx by PROCLIB=ddname.
 - The full procedure library ddname is specified with PROCLIB=.
 - Not just using a suffix as with //*MAIN PROC=xx.
 - JES3 will support procedure library ddnames IATPLBxx and PROCnn (JES3 and JES2 naming conventions).
 - IATPLBxx and PROCnn must be defined in the initialization stream or JES3 procedure.
 - PROCnn cannot be specified using //*MAIN PROC=xx as the suffix is expanded to ddname of IATPLBxx.

JES3 JCL support – PROCLIB=

Defining procedure libraries



- JES3 V2R2 will support PROCLIB ddnames IATPLBxx and PROCnn.

- Can be defined using DYNALLOC initialization statements:

```
DYNALLOC, DDN=IATPLB04, DSN=FREDS.PROCLIB
```

```
DYNALLOC, DDN=IATPLB04, DSN=DEPT.PROCLIB
```

```
DYNALLOC, DDN=IATPLB04, DSN=AREA.PROCLIB
```

```
DYNALLOC, DDN=PROC22, DSN=MYJES2.PROCLIB
```

- Can be defined with DD JCL statements in the JES3 start procedure:

```
//IATPLBST DD DISP=SHR, DSN=SYS1.PROCLIB
```

```
//IATPLBPI DD DISP=SHR, DSN=PICKLE.PROCLIB
```

```
//PROC49 DD DISP=SHR, DSN=DEPT49.PROCLIB
```

JES3 JCL support – PROCLIB=

Other JES3 proclib references



- *INQUIRY,PROCLIB updated with DDNAME=.
 - When DDNAME=ddname is used, the full ddname of the procedure library must be specified.
 - Examples: IATPLB04, PROC49
 - When ID=xx is used, IATPLBxx form of the proclib ddname is used.
 - When neither ID= and DDNAME= are used, then all proclibs are returned.
- All other existing JES3 references using the 2-character id 'xx' will continue to identify a procedure library with ddname IATPLBxx:
 - STANDARDS JES3 initialization statement for INTPROC=xx, STCPROC=xx, and TSOPROC=xx.
 - *I,PROCLIB,ID=xx (JES3 command)
 - //*MAIN PROC=xx (JES3 JECL statement)

JES3 JCL support

// OUTPUT DDNAME=



- **DDNAME=** specifies DD statements to which OUTPUT specification is applied.
 - Similar to DDNAME= on `//*FORMAT JECL` statement.
 - DDNAME= *ddname*, *procstepname.ddname*, *stepname.ddname* or *stepname.procstepname.ddname*
 - Provides a direct reference from the OUTPUT statement to one or more sysout data sets.
 - OUTPUT statement(s) applied to a data set will be the first of:
 - 1) Direct OUTPUT statements identified using DD OUTPUT= parameter.
 - 2) Step level OUTPUT statements with DDNAME= match
 - New capability
 - 3) Job level OUTPUT statements with DDNAME= match
 - New capability
 - 4) Step level OUTPUT statements with DEFAULT=YES parameter.

JES3 JCL support

DDNAME= example



```
//OUTDDNAM JOB      MSGCLASS=T,MSGLEVEL=(1,1),CLASS=A
//DFLT1     OUTPUT  DEST=DESTD,DEFAULT=YES
//DSTJ1     OUTPUT  DEST=DSTJ1,DDNAME=SYSUT2
//DSTJ2     OUTPUT  DEST=DSTJ2,DDNAME=STEP2.SYSUT2
//*
//STEP1     EXEC    PGM=IEBGENER
//SYSPRINT  DD      SYSOUT=(,)
//SYSIN     DD      DUMMY
//SYSUT2    DD      SYSOUT=(,)
//SYSUT1    DD      *
INSTREAM DATA FOR STEP1
//*
//STEP2     EXEC    PGM=IEBGENER
//SYSPRINT  DD      SYSOUT=(,)
//SYSIN     DD      DUMMY
//SYSUT2    DD      SYSOUT=(,)
//SYSUT1    DD      *
INSTREAM DATA FOR STEP2
//*
//STEP3     EXEC    PGM=IEBGENER
//DSTS3     OUTPUT  DEST=DSTS3,DDNAME=SYSUT2
//SYSPRINT  DD      SYSOUT=(,)
//SYSIN     DD      DUMMY
//SYSUT2    DD      SYSOUT=(,),OUTPUT=(*.DSTJ2)
//SYSUT1    DD      *
INSTREAM DATA FOR STEP3
//
```

- See next slide for explanations

JES3 JCL support

DDNAME= example explanation



- STEP1.SYSUT2 will have DEST=DSTJ1 due to a DDNAME= match with statement DSTJ1.
- STEP2.SYSUT2 will have DEST=DSTJ2 due to a DDNAME= match with statement DSTJ2.
 - DSTJ1 is not used because DSTJ2 has more matching qualifiers.
 - If DSTJ2 had only DDNAME=SYSUT2, then one instance with DEST=DSTJ2 and one instance with DEST=DSTJ1 would be created.
- STEP3.SYSUT2 will have DEST=DSTJ2 due to the direct reference OUTPUT=(*.DSTJ2).
 - Even though DSTS3 at the step level and DSTS1 at the job level could both match, the direct reference with OUTPUT=(*.DSTJ2) means no search for a DDNAME= match at the step or job level will be done.
- All other sysout will have DEST=DESTD due to the default DFLT1.

JES3 JCL support

// OUTPUT MERGE=YES



- **MERGE=YES** indicates OUTPUT specification defines base values used for all SYSOUT at the job or step level.
 - Same as existing MERGE text unit support.
 - Does not create additional instances of the data sets.
 - Other OUTPUT statements are applied normally and may create additional instances of the data sets.
 - Only one MERGE=YES statement per context is used.
 - First MERGE=YES at the step or job level used.

JES3 JCL support

MERGE=YES example



```
//OUTDDMRG JOB      MSGCLASS=T,MSGLEVEL=(1,1),CLASS=A
//DFLT1     OUTPUT  DEST=DESTD,DEFAULT=YES
//MERGE1    OUTPUT  DEST=DEST1,FORMS=FORM01,MERGE=YES
//MERGE2    OUTPUT  DEST=DEST2,FORMS=FORM02,MERGE=YES
// *
//STEP1     EXEC    PGM=IEBGENER
//SYSUDUMP  DD      SYSOUT=(,)
//SYSPRINT  DD      SYSOUT=(,)
//SYSIN     DD      DUMMY
//SYSUT2    DD      SYSOUT=(,)
//SYSUT1    DD      *
INSTREAM DATA FOR STEP1
/*
//STEP2     EXEC    PGM=IEBGENER
//DFLTS2    OUTPUT  FORMS=FORMD2,DEFAULT=YES
//MERGES2   OUTPUT  DEST=DESTM2,MERGE=YES
//SYSUDUMP  DD      SYSOUT=(,)
//SYSPRINT  DD      SYSOUT=(,)
//SYSIN     DD      DUMMY
//SYSUT2    DD      SYSOUT=(,),OUTPUT=(*.MERGE2)
//SYSUT1    DD      *
INSTREAM DATA FOR STEP2
//
```

- See next slide for explanations

JES3 JCL support

MERGE=YES example explanation



- At the JOB level, only OUTPUT statement **MERGE1** will be used as a base.
 - MERGE2 is a valid OUTPUT statement that can be directly referenced.
- Within **STEP1** there is no OUTPUT statement with MERGE=YES, so OUTPUT statement **MERGE1** will be used as a base for all SYSOUT data sets in STEP1.
 - There are no direct references to OUTPUT statements, so the default OUTPUT statement **DFLT1** will be used to update the base.
 - The result is that each SYSOUT data set will have DEST=DESTD and FORMS=FORM01.
- Within **STEP2** the OUTPUT statement **MERGES2** with MERGE=YES will be used as the base for all SYSOUT data sets in STEP2.
 - SYSUT2 has a direct reference to OUTPUT statement MERGE2 which will be used to update the base.
 - The result is that SYSUT2 will have DEST=DEST2 and FORMS=FORM02.
- Also within **STEP2** we have another default OUTPUT statement **DFLTS2** that will be used to update the base for the remaining SYSOUT data sets in STEP2.
 - The result is that each remaining SYSOUT data set will have DEST=DESTM2 and FORMS=FORMD2.

Tracking JECL usage

- Generic Tracker macro GTZTRACK can be used to track JES3 JECL statements in a job stream during input service.
 - GTZ tracking occurs only when enabled by command:
SETGTZ TRACKING=ON
- For most jobs, a single GTZ record is created that summarizes the JES3 JECL statements used for the job stream.
 - Generated in module IATISLG – Input service logic driver that reads and parses JCL statements. Most JES3 JECL statements are recognized here.
- Occasionally a second GTZ record may be created for JCL which is processed by a card, disk, or tape reader.
 - Generated in module IATISRI – Common reader routine. Tracking will cover JES3 JECL statements which are not passed to IATISLG.

Tracking JECL usage

JECL tracked



- Tracked by IATISLG
 - `//*DATASET`
 - `//*FORMAT`
 - `//*MAIN`
 - `//*NET`
 - `//*NETACCT`
 - `//*OPERATOR`
 - `//*PROCESS`
 - `//*ROUTE`
 - `//*SIGNON`
- Tracked by IATISRI
 - `/**command` (JES3 command statement)
 - `//*PAUSE`
- Not tracked
 - `//*ENDDATASET` – valid only when `//*DATASET` is used
 - `//*ENDPROCESS` – valid only when `//*PROCESS` used
 - `//*SIGNOFF` – JES3 usage compatible with JES2.

Tracking JECL usage

GTZ record



- GTZ records include a 64 byte Event Description field (EVENTDESC) the contents of which GTZTRACK user can define.
 - Also include a owner id = 'IBMJES3'
 - Also include a source id = 'IATISLG' or 'IATISRI'
- For JES3 JECL tracking records the Event Description field depends upon the source: IATISLG or IATISRI.
 - Both contains flags which identify the JECL statements found.
 - Both include the port of entry (RQPOE) for the job.
 - For IATISLG the job name and submitter user ID are included.
 - For IATISRI the job name last job card read and the reader's DDNAME are included.

Tracking JECL usage

GTZ record – flags

- Layout of 19 characters that contain the flags which identify the JECL statements found:
`' | 00000000 000xxxxx | '`
 - First and last '|' characters and the blank are delimiters established for display purposes when the records are retrieved.
 - 'xxxxxx' are unused flags.
 - '00000000 000' are set to '1' for each tracked JECL statement:
 - JES3 command statement
 - /*NET
 - /*PROCESS
 - /*DATASET
 - /*NETACCT
 - /*ROUTE
 - /*FORMAT
 - /*OPERATOR
 - /*SIGNON
 - /*MAIN
 - /*PAUSE
- See *z/OS V2R2 MVS Diagnosis: Tools and Service Aids*
 - Chapter 11. The generic tracker facility
 - New section for “JES3 control statement tracking”

Tracking JECL usage

GTZ record example



- GTZ tracking records can be displayed with command:
D GTZ,TRACKDATA=(OWNER=IBMJES3)
- Example of jobs submitted from IBMUSER:

```
-----  
INSTANCE:          1                      COUNT:          1  
EVENTDESC:         '|00110011 000xxxxx| INTRDR    CRJOB001 IBMUSER '  
OWNER:             IBMJES3                SOURCE:          IATISLG  
EVENTDATA:         x000000000000000000    x000000000000000000  
PROGRAM:           *UNKNOWN               PROGRAMOFFSET: x000000000000000000  
HOMEJOB:           JES3                   HOMEASID:          x001F  
EVENTJOB:          JES3                   EVENTASID:        x001F  
AUTHORIZED:        YES                    FIRST TIME:        2015-07-21 11:46:16  
-----  
INSTANCE:          2                      COUNT:          1  
EVENTDESC:         '|00110010 000xxxxx| INTRDR    CRJOB0X1 IBMUSER '  
OWNER:             IBMJES3                SOURCE:          IATISLG  
EVENTDATA:         x000000000000000000    x000000000000000000  
PROGRAM:           *UNKNOWN               PROGRAMOFFSET: x000000000000000000  
HOMEJOB:           JES3                   HOMEASID:          x001F  
EVENTJOB:          JES3                   EVENTASID:        x001F  
AUTHORIZED:        YES                    FIRST TIME:        2015-07-21 11:46:16  
-----
```

Tracking JECL usage

GTZ record example

- Example of same job CRJOB001 submitted through a card reader:
 - Note that //*PAUSE was tracked in IATISRI rather than IATISLG.

```
-----  
INSTANCE:          1                                COUNT:          3  
EVENTDESC:         '|00000001 000xxxxx| RDR011    CRJOB001 RDR011  '  
OWNER:             IBMJES3                          SOURCE:          IATISRI  
EVENTDATA:         x000000000000000000            x000000000000000000  
PROGRAM:           *UNKNOWN                        PROGRAMOFFSET: x000000000000000000  
HOMEJOB:           JES3                            HOMEASID:          x001F  
EVENTJOB:          JES3                            EVENTASID:         x001F  
AUTHORIZED:        YES                             FIRST TIME:        2015-07-21 12:24:32  
-----  
INSTANCE:          2                                COUNT:          3  
EVENTDESC:         '|00110010 000xxxxx| RDR011    CRJOB001      '  
OWNER:             IBMJES3                          SOURCE:          IATISLG  
EVENTDATA:         x000000000000000000            x000000000000000000  
PROGRAM:           *UNKNOWN                        PROGRAMOFFSET: x000000000000000000  
HOMEJOB:           JES3                            HOMEASID:          x001F  
EVENTJOB:          JES3                            EVENTASID:         x001F  
AUTHORIZED:        YES                             FIRST TIME:        2015-07-21 12:24:33  
-----
```

JES3 scaling

- Various JES3 control blocks are moved from 2GB data spaces.
 - Limited the number of control blocks that can be created which limits the number of jobs and spool files that can concurrently exist.
 - Existing limits have never been reported as an issue, but we also do not know how close to the limits customers may be getting.
- JES3 cell pool support is updated to support cell pools in 64-bit addressable storage (above the bar).
 - Cell pools for selected control blocks are moved to above the bar.
 - No longer limited to 2GB of storage.
- Customers can control the number of control blocks that are to be allocated.
- New JES3 Health Checks monitor the control block usage and alert customers to potential shortages.

JES3 scaling

Control blocks affected

- Control blocks moved to above the bar:
 - DOT = Dataset Output Table (IATYDOT)
 - JET = JDS Entry Table (IATYJET)
 - OST = OSE Summary Table (IATYOST)
 - SEE = SAPI Exclusion Element (IATYSEE)
- Also moved the in storage copy of the JCT to above the bar storage.
 - Added a JCTX extension to allow for additional job data on spool for current and future enhancements

JES3 scaling

New OPTIONS keywords



- New keywords added to the OPTIONS initialization statement to allow customers to specify the size of a pool by number of cells.
 - DOTPOOL=(*primary*,*total*) – SEEPOOL=(*primary*,*total*)
 - JETPOOL=(*primary*,*total*) – OSTPOOL=(*primary*,*total*)
- *primary* and *total* are both optional parameters
 - *primary* specifies the number of cells in the primary extent.
 - *total* specifies the total number of cells that can be created.
 - Values represent 1000s of cells.
 - For example, JETPOOL(640) represents 640,000 cells.
 - Default and maximum values vary due to a number of factors such as the size of the control blocks and number of jobs allowed on the system.

JES3 scaling

*INQUIRY,C updated

- Display buffer pools command has been updated to display the status of the DOT, JET, OST and SEE cell pools:

*INQUIRY,C,*cpid*[,U[N=*nn*]]

- *cpid* is the cell pool id DOT, JET, OST or SEE
- U displays a list of the largest users of the specified cell pool.
- N= specifies the number of users displayed (default=10, max=100)

- Example:

```
*i,c,jet
```

```
IAT8727 JET          CELLPOOL USAGE   <1% OF MAXIMUM.....696,303
IAT8728 CURRENT NUMBER IN USE .....21
IAT8729 MAXIMUM NUMBER USED .....157
IAT8735 CURRENT SECONDARY EXTENTS IN USE .....0
IAT8736 MAXIMUM SECONDARY EXTENTS USED .....0
```

JES3 scaling

*INQUIRY,OPTIONS updated



- *I,OPTIONS command displays the information specified or defaulted to on the OPTIONS initialization statement.
 - Updated to include the cell pool keywords DOTPOOL, JETPOOL, OSTPOOL, and SEEPOOL.
 - Displays only values specified on the OPTIONS statement.
 - If no value is specified (i.e. default) then no value is displayed.
 - Defaults are not a fixed value and are determined during JES3 initialization. Use *I,C,cpid to see current values.
- Example (all are valid syntax for the OPTIONS statement):

```
*i,options
IAT8646 OPTIONS INQUIRY RESPONSE
. . .
DOTPOOL=(2000,) , JETPOOL=(0640,10000) ,
OSTPOOL=(,30000) , SEEPOOL=(,)
OPTIONS INQUIRY RESPONSE COMPLETE
```

New JES3 health checks



- Needed a mechanism to warn customers of high utilization of critical JES3 resources such as control blocks.
- Using health checks provides a number of advantages:
 - Existing mechanism!
 - Checks can run outside of JES3 address space.
 - All currently run on the JES3 global in the Health Checker.
 - Constant monitoring can occur allowing for warnings to come and go as resource utilization changes (i.e. not just a one time warning).
 - Can have dynamic message severity levels based upon utilization.
 - For example, 60% utilization generates a low severity message, 70% a medium severity message, and 85% a high severity message.
 - Health checker displays the messages with different characteristics depending upon the severity.

New JES3 health checks

Checks and parameters



- `CHECK (IBMJES3, JES3_DOT_POOL_USAGE) ,`
`INTERVAL (00:10) ,`
`PARM (' THRESHOLD_HIGH (85%) , THRESHOLD_MED (70%) ' )`
 - `INTERVAL (00:10)` specifies the check runs every 10 minutes.
 - Parameter `THRESHOLD_HIGH (85%)` specifies high message severity when the DOT cell pool utilization is 85% or more.
 - Parameter `THRESHOLD_MED (70%)` specifies medium message severity when the DOT cell pool utilization is 70% or more.
- `CHECK (IBMJES3, JES3_JET_POOL_USAGE) ,`
`INTERVAL (00:10) ,`
`PARM (' THRESHOLD_HIGH (85%) , THRESHOLD_MED (70%) ,`
`THRESHOLD_LOW (60%) ' )`
 - Parameter `THRESHOLD_LOW (60%)` specifies low message severity when the JET cell pool utilization is 60% or more.

New JES3 health checks

Checks and parameters



- `CHECK (IBMJES3, JES3_OST_POOL_USAGE) ,
INTERVAL (00:10) ,
PARM (' THRESHOLD_HIGH (85%) , THRESHOLD_MED (70%) ,
THRESHOLD_LOW (60%) ')`
- `CHECK (IBMJES3, JES3_SEE_POOL_USAGE) ,
INTERVAL (00:10) ,
PARM (' THRESHOLD_HIGH (85%) , THRESHOLD_MED (70%) ')`
- Other notes:
 - The `THRESHOLD_MED (70%)` value is equivalent to the `THRESHOLD=` value used by JMF to generate warning.
 - We specify a `THRESHOLD_LOW (60%)` value for the JET and OST pool utilization as they are the cell pools that can potentially be heavily utilized.
 - Want to give customers an earlier heads up.

New JES3 health checks

Checks and parameters



- `CHECK (IBMJES3, JES3_DATASET_INTEGRITY) ,
INTERVAL (ONETIME) , SEVERITY (LOW) ,
PARM ('DSI (YES) ')`
 - `INTERVAL (ONETIME)` specifies the check runs once.
 - Parameter `SEVERITY (LOW)` specifies low message severity.
 - Parameter `PARM ( 'DSI (YES) ' )` specifies that a message is generated when DSI is not specified for the JES3 entries in the Program Properties table (PPT).
 - Can be changed to `PARM ( 'DSI (NO) ' )` if you choose to not use DSI.

Toleration

- JES3 release toleration APAR OA43563 is required for systems in the JESplex which are at JES3 V2R1 or V1R13.
 - Tolerates parameters added in V2R2, but does not roll back the support added in V2R2.
 - For example Data Set Integrity is effective only with JES3 V2R2 as a global.
 - See APAR/PTFs for details.



Thank You!

Session 17600



Complete your session evaluations online at www.SHARE.org/Orlando-Eval

8/13/2015

