IBM education Assistant (IEA) for z/OS V2R3

JES2: Job Execution Control Phase II

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Migration & Coexistence Considerations
- Installation
- Session Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.
- Additional Trademarks:
  - None.

Session Objectives

- In this session we will introduce new JES2 Job control enhancements implemented in JES2 V2R3.
  - The use of BEFORE=/AFTER=/DELAY= on the SCHEDULE statement (i.e. - ad hoc job sequencing).
  - JES2 processing of JES3 //*NET statements (using enhanced version of JES2 JOBGROUPs).
  - Support the OUTPUT JCL card for JOBGROUP logging jobs.
  - Support NOTIFY= on JOBGROUP card and NOTIFY JCL card for JOBGROUP logging jobs.

Overview – SCHEDULE enhancements

- Problem Statement / Need Addressed
  - The need to have a method to implement simple ad hoc job sequencing operations without creating/accessing a static JOBGROUP. This is similar to certain customer exits.
- Solution
  - Provide SCHEDULE BEFORE=/AFTER=/DELAY=, similar in operation to common customer exits.
- Benefit / Value
  - Easier migration of customer exits to native JES2 function.

Overview – JES3 //*NET support

- Problem Statement / Need Addressed
  - Provide further assistance to JES3 customers that wish to migrate to JES2.
- Solution
  - If switched on (via command), JES2 will migrate JES3 //*NET statements to JES2 using a modified version of JES2 Job Execution Controls (JEC).
- Benefit / Value
  - Easier migration from JES3 to JES2.
  - Ability to use //*NET statements in JES2

Overview – JOBGROUP //OUTPUT card

- Problem Statement / Need Addressed
  - Cannot specify characteristics for job group logging job output.
- Solution
  - New support added for //OUTPUT cards in a JOBGROUP.
- Benefit / Value
  - Allows specification of characteristics of JOBGROUP logging job's output JOE.
  - Allows output more consistent with standard job output.

Overview – JOBGROUP NOTIFY

- Problem Statement / Need Addressed
  - Current JOBGROUP processing does not support NOTIFY functionality.
- Solution
  - New support added to allow NOTIFY= on JOBGROUP card.
    - Works like NOTIFY= on JOB card.
  - New support added to allow JOBGROUPs to support the new NOTIFY JCL card (i.e. - notify via email address).
- Benefit / Value
  - Allows NOTIFY specification consistent with standard jobs.

Usage & Invocation – SCHEDULE additions (1)

- The SCHEDULE statement will now accept BEFORE=, AFTER=, and DELAY= keywords.
- Facilitates ad hoc sequencing of jobs without accessing a static JOBGROUP.
  - Also known as 'Dynamic Job Sequencing”
- Mutually exclusive with static JOBGROUPs.
  - Cannot specify BEFORE=, AFTER, or DELAY= if JOBGROUP= is specified.
  - Only one BEFORE= and one AFTER= allowed per job.
- Implemented using logic in job selection ($QGET) and job transition ($QMOD).

Usage & Invocation – SCHEDULE additions (2)

- Syntax example of the new keywords :

```
//*-------------------------------------------------------
//JOBC     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//         SCHEDULE BEFORE=JOBE,AFTER=JOBA,DELAY=YES
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
```

  - BEFORE= indicates the job containing the SCHEDULE statement must run before BEFORE=jobname.
    - In our example, JOBC must run before JOBE.
  - AFTER= indicates the job containing the SCHEDULE statement must run after AFTER=jobname.
    - In our example, JOBC must run after JOBA.
  - In our example above, the job sequence would be :
    - JOBA->JOBC->JOBE

Usage & Invocation – SCHEDULE additions (3)

- Dynamic Job Sequencing issues :
  - The SCHEDULE statement is processed at conversion time.
  - Conversion delays could allow jobs to execute prematurely, causing inconsistent results.
  - Therefore, there is a need to delay jobs until the entire 'set' can be recognized by the system.
    - HOLDUNTL is enhanced to allow 'second' granularity.
      - `// SCHEDULE HOLDUNTL=('+hh:mm:ss')`
      - Allows short delays before further job selection criteria (such as BEFORE= and AFTER=) are evaluated.
    - DELAY=YES added for dependent jobs that do not have an AFTER=.
      - AFTER= is considered an implied DELAY=YES.
      - Indicates that there is at least one parent job.
        - Job will not run until all 'parent' jobs are complete.

Usage & Invocation – SCHEDULE additions (4)

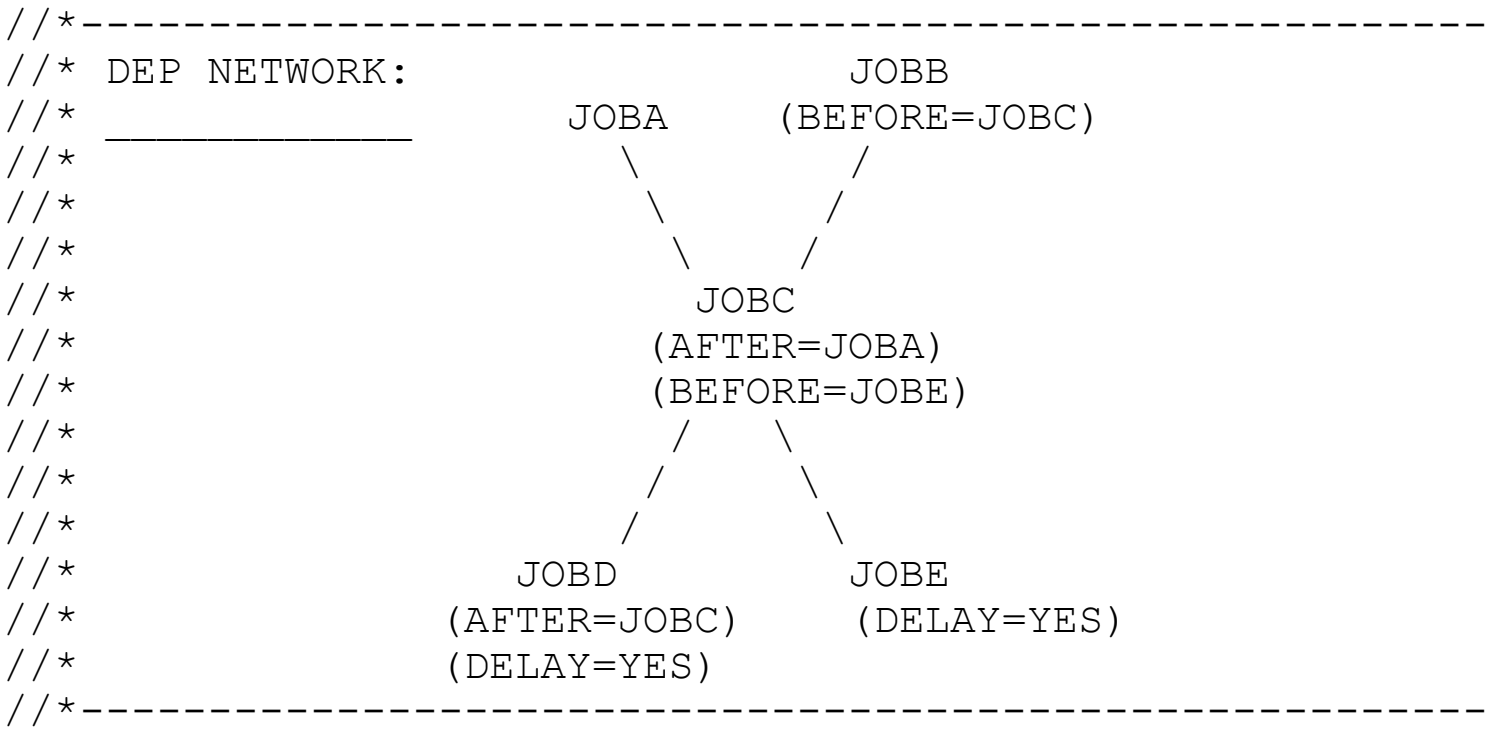
- Dynamic Job Sequencing – additional information :
  - Implemented entirely via additional fields in the job ($JQE).
    - No JOBGROUP and associated structures are created/maintained.
      - No 'logging job' JQE is created.
      - No 'ZJC CTENT' structures are created.
    - One downside is performance.
      - Job queue scans are needed (not so for JOBGROUPs).
      - Recommended to create a JOBGROUP if possible.
    - Another difference is no checks for how parent job ends.
  - BEFORE=, AFTER=, DELAY= information is only returned on a per-job basis via $DJ.
    - Cannot use $DG table-based commands.
    - Another reason it is recommended to create a JOBGROUP for complex networks.

Usage & Invocation – SCHEDULE additions (5)

- Dynamic Job Sequencing – Extended Status SSI support :
  - No JOBGROUP implies that no JOBGROUP sections will be returned.
  - New filters added for SCHEDULE BEFORE/AFTER/DELAY
  - New STATDYND terse section added :
    - BEFORE= job name (STDYBEJB)
    - AFTER= job name (STDYAFJB)
    - Job is delayed due to dynamic dependency flag (STDY1DLY)
      - STDY1DLY=ON implies that the job is currently delayed due to a DELAY=YES or AFTER=.

Usage & Invocation – SCHEDULE additions (6)

- Dynamic Scheduling example - Overview :



- Note that this is only one of many ways to specify the same job sequencing 'network'.

Usage & Invocation – SCHEDULE additions (7)

- Dynamic Scheduling example - JCL :

```
//*-------------------------------------------------------
//JOBE     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//         SCHEDULE DELAY=YES
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
//JOBD     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//         SCHEDULE AFTER=JOBC,DELAY=YES
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
//JOBC     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//         SCHEDULE BEFORE=JOBE,AFTER=JOBA
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
//JOBB     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//         SCHEDULE BEFORE=JOBC
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
//JOBA     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------
```

Usage & Invocation – //*NET support (1)

- Support must first be enabled via command.
  - **`$t inputdef,jes3jecl=process`**
  - **`$t jecldef,jes3=(net=process)`**
- Once enabled, JES2 will migrate JES3 //*NET JECL statements to JES2 JOBGROUPs.
  - JES3 specific functions are ignored (i.e. - DEVPOOL=, DEVRELSE=, RELSCHCT=).
  - Created JOBGROUPs are marked as having a //*NET statement origin.
    - Done so JOBGROUP run time processing can match the behavior of JES3 //*NET.
    - //*NET JOBGROUPs are processed like JES3 NETs.
      - Run time behavior different from standard JOBGROUPs.
  - JOBGROUP name is the NETID= as specified on the //*NET statement.
    - Name space is shared with traditional JOBGROUPs.

Usage & Invocation – //*NET support (2)

- The intent is to support JES3 //*NET behavior as closely as possible.
  - HOLD counts are maintained by the //*NET JOBGROUP.
  - Commands and WTOs to modify HOLD counts are supported (see below).
    - Similar to functionality in JES3.
  - Run time behavior for //*NET JOBGROUPs is tailored to simulate JES3 //*NET behavior as much as possible.
    - Substantial run time differences with traditional JOBGROUP behavior

Usage & Invocation – //*NET support (3)

- //*NET support – Extended Status SSI support :
  - JES2 JOBGROUPs are used to implement //*NET networks.
  - Therefore, the same SSI sections are returned.
    - RELEASE= job name list is returned as multiple dependency (STATDB) objects.
  - New job information subsection added (STATNETI).
    - Contains //*NET statement keyword information (original HOLD count, NORMAL, ABNORMAL, ABCMP, NRCMP, OPHOLD, etc...).
  - Updated Job's JOBGROUP section (STATJZXC) with :
    - Current HOLD count (STJZCHLD)
    - NETREL= NETID name (STJZNRID)
    - NETREL= Job name (STJZNRJB)
  - HOLD count filter added (STATHCFV).
    - Allows filtering on HOLD counts =, >, <, >=, <=, != to STATHCFV.

Usage & Invocation – //*NET support (4)

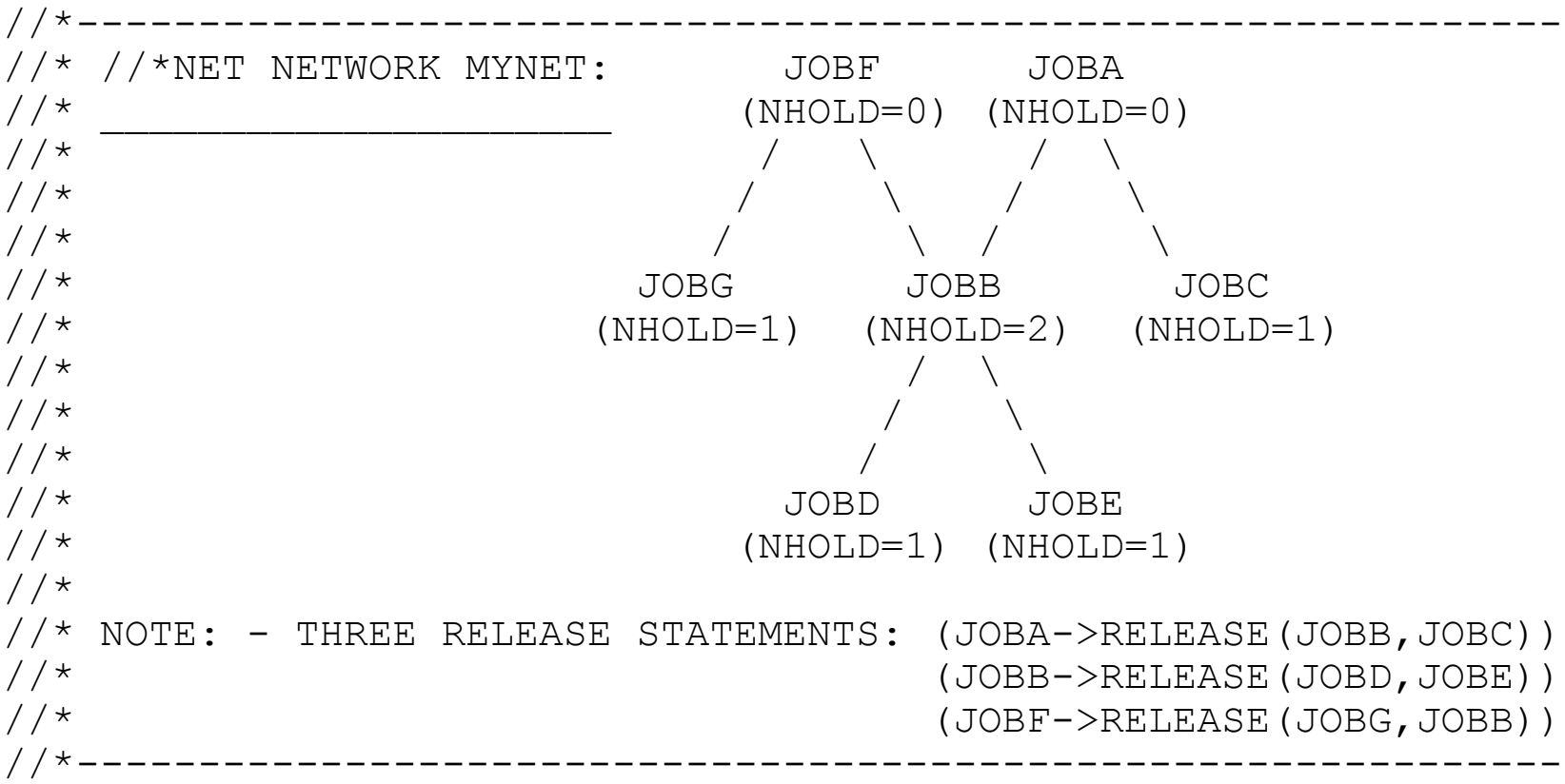
- //*NET support - Commands :
  - Enable JES2 to process //*NET statements :
    - `$t inputdef,jes3jecl=process`
    - `$t jecldef,jes3=(net=process)`
  - Additional commands for jobs associated with an //*NET JOBGROUP.
    - Display HOLD count value:
      - `$DJQ,JM=MYJOB,NHOLD`
    - Decrement HOLD count value:
      - `$TJQ,JM=MYJOB,NHOLD=-`
    - Increment HOLD count value:
      - `$TJQ,JM=MYJOB,NHOLD=+`
  - Standard JES2 JOBGROUP commands can also be used.

Usage & Invocation – //*NET support (5)

- //*NET support – Commands (2) :
  - Display //*NET JOBGROUP (via existing JOBGROUP commands) :
    - Overview of a //*NET JOBGROUP :
      - `$DG*,JM=MYNET`
    - Display jobs in a //*NET JOBGROUP :
      - `$DG*,JM=MYNET,JOBS`
    - Display dependencies in a //*NET JOBGROUP :
      - `$DG*,JM=MYNET,DEP`
  - Notes :
    - MYNET is the NETID name (i.e. - //*NET NETID=MYNET) .
    - Dependencies are created from the //*NET RELEASE=(jobname[,jobname]...) clause.

Usage & Invocation – //*NET support (6)

- //*NET support Example - Overview :



Usage & Invocation – //*NET support (7)

- //*NET support Example - JCL :

```
//*-------------------------------------------------------------
//JOBA     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,RELEASE=(JOBB,JOBC),NHOLD=0
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBB     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,RELEASE=(JOBD,JOBE),NHOLD=2
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBC     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,NHOLD=1
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBD     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,NHOLD=1
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBE     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,NHOLD=1
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBF     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,RELEASE=(JOBG,JOBB),NHOLD=0
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
//JOBG     JOB TIME=NOLIMIT,REGION=0K,MSGCLASS=A,CLASS=A
//*NET     NETID=MYNET,NHOLD=1
//STEP1    EXEC PGM=IEFBR14
//*-------------------------------------------------------------
```

Usage & Invocation – JOBGROUP OUTPUT card

- New support added for //OUTPUT cards in a JOBGROUP.
  - JESDS= or MERGE=YES must be specified to be effective
    - Not DEFAULT=YES (only JOBGROUP output is are JESDS)
- Implies the JOBGROUP 'logging job' will go through Converter/Interpreter.
  - A dummy job is passed through normal conversion phase.
  - No JES2 exits are called.
  - Interpreter always runs (in JES2 address space if needed).
- This results in output that is consistent with output from standard jobs.
  - Has all three data sets (JESMSGLG, JESJCL, JESYSMSG).

Usage & Invocation – JOBGROUP NOTIFY

- Covered under “JES2: email enhancements” presentation.

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - Any JES2 user.

Migration & Coexistence Considerations

- From JES2 z/OS 2.1 or z/OS 2.2
  - APAR OA48299 and OA53537 needed on z/OS 2.1 or z/OS 2.2 member to coexist in a MAS with z/OS 2.3
  - APAR OA48299 is also highly recommended for fall back as well
    - Some new data structures created by z/OS 2.3 JES2 may result in problems if OA48299 is not installed.
- //*NET support requires z22 checkpoint level.

Installation

- None

Session Summary

- In this session we discussed the following JES2 enhancements :
  - The use of BEFORE=/AFTER=/DELAY= on the SCHEDULE statement (i.e. - ad hoc job sequencing).
  - JES2 processing of JES3 //*NET statements (using enhanced version of JES2 JOBGROUPs).
  - Support of the OUTPUT JCL card for JOBGROUP logging jobs.
  - Support of NOTIFY= on JOBGROUP card and NOTIFY JCL card for JOBGROUP logging jobs.

Appendix

- Publications
  - z/OS V2R3.0 JES Application Programming – SA32-0987-30
  - z/OS V2R3.0 JES2 Commands – SA32-0990-30
  - Z/OS V2R3.0 JES2 Diagnosis - GA32-0993-30
  - z/OS V2R3.0 JES2 Initialization and Tuning Guide – SA32-0991-30
  - z/OS V2R3.0 JES2 Initialization and Tuning Reference – SA32-0992-30
  - z/OS V2R3.0 JES2 Installation Exits – SA32-0995-30
  - z/OS V2R3.0 JES2 Macros – SA32-0996-30
  - z/OS V2R3.0 JES2 Messages – SA32-0989-30
  - z/OS V2R3.0 MVS JCL Reference - SA23-1385-30
  - z/OS V2R3.0 MVS Using the Subsystem Interface – SA38-0679-30