IBM Education Assistance (IEA) for z/OS V2R3

Line Item Name: GDG Bias
Element/Component: MVS Device Allocation

Agenda

- Trademarks
- Session Objectives
- Overview
- Usage & Invocation
- 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 presentation, we will describe the functionality of the new GDGBIAS support and how to use it, including:
  - JCL and restart processing
  - Examples of use
  - JES configuration and definitions
  - Programming interfaces

Overview

- Problem Statement / Need Addressed
  - The association between a relative generation number and an absolute generation number of a generation data group (GDG) within a job is traditionally established for the life of the job. Although this relationship is intuitive, it has some limitations:
    - Some types of restarts require JCL modifications so that the restart job refers to the correct physical data set.
    - Some customers have implemented their own USERMODs to associate the relative generation number and absolute generation number on a step basis, rather than a job basis.
- Solution
  - z/OS will now allow jobs to manage references to generation data sets on a step or a job basis.
- Benefit / Value
  - JCL can be modified so that restarts will work without modification.
  - Customers no longer need to have custom USERMODs to implement this function.

Usage & Invocation

- A new JCL keyword on the JOB statement will be provided to select GDG behavior for the job: GDGBIAS=JOB|STEP
  - GDGBIAS=JOB will request the “traditional” behavior of associating relative and absolute generation data set numbers on a job basis.
  - GDGBIAS=STEP will request the behavior of associating relative and absolute generation data set numbers on a step basis.
- A similar job class attribute will be provided in both JES2 and JES3.
  - The job class attribute only applies when the GDGBIAS keyword is not coded on the JOB statement.
  - The job class attribute defaults to GDGBIAS=JOB.
    - By default, those customers who do not have a custom USERMOD should not see any change in behavior.
- GDGBIAS is determined at a job level.
  - There is no ability to mix step basis and job basis in the same job, even for different GDGs.

Usage & Invocation

- To illustrate how this works, here is a simple example. Assume that GDG01 is a GDG with no existing generations.

```
//JOB1 JOB GDGBIAS=JOB
//STEP1 EXEC PGM=IEFBR14
//DD01 DD DSN=GDG01(+1),DISP=(NEW,CATLG)
//STEP2 EXEC PGM=IEFBR14
//DD02 DD DSN=GDG01(+1),DISP=OLD
```

  Since GDGBIAS=JOB is in effect, STEP1 creates a new data set GDG01.G0001V00. STEP2 refers to the same data set G0001V00.

- To get the same behavior with GDGBIAS=STEP:

```
//JOB2 JOB GDGBIAS=STEP
//STEP1 EXEC PGM=IEFBR14
//DD01 DD DSN=GDG01(+1),DISP=(NEW,CATLG)
//STEP2 EXEC PGM=IEFBR14
//DD02 DD DSN=GDG01(0),DISP=OLD
```

- Obviously, it is necessary to modify JCL beyond just adding the GDGBIAS keyword.

- See the Appendix for more complicated examples.

Usage & Invocation

- Internally, Allocation maintains a list of the GDG's used within a job, and the relationship between the (0) relative generation number and the corresponding absolute generation number, in a control block called the GDGNT.
- When GDGBIAS=STEP is in effect, each job step gets an “empty” GDGNT.
  - The first reference to a GDG in a step will cause the system to retrieve the current state of the GDG from the Catalog, and add the current relationship between the (0) relative generation number and the corresponding absolute generation number to the GDGNT.
  - Subsequent references to that GDG in the same step will use the relationship defined in the GDGNT.
  - If a generation data set is not cataloged, the relationship is unchanged. For example, if a step creates a new (+1) generation with DISP=(NEW,PASS), a subsequent step that catalogs it with DISP=(OLD,CATLG) must still refer to it as (+1).

Usage & Invocation

- GDG-All considerations:
  - GDG-All requests, where a relative generation number is not specified, so that all generations are allocated, does not use the GDGNT for GDGBIAS=JOB or STEP.
- Dynamic allocation considerations:
  - By default, dynamic allocation uses the same GDGNT, and therefore the same relationship between relative and absolute generation numbers, as batch JCL. Therefore, GDGBIAS=STEP will cause dynamic allocation to resolve any relative generation numbers on a step basis, just as in JCL.
  - The existing S99GDGNT option for dynamic allocation will cause the GDGNT to be ignored and resolve the generation using the Catalog.

Usage & Invocation

- Restart considerations:
  - When using GDGBIAS=STEP, the system resolves the (0) level of the GDG using the Catalog at the first reference for each step.
  - When using GDGBIAS=STEP, it should not be necessary to change the relative generation number in the job as long as the state of the GDG has not changed.
    - If another job created a new generation before restarting the failing job, the failing job would resolve generation numbers based on the newly updated Catalog information. In this case, it still may be necessary to update the JCL before restarting.
    - If the failing job changed the state of the GDG, that may also require a JCL change before restarting. As an example, if you create a new (+1) generation, with DISP=(NEW,CATLG,CATLG), and the job failed in that step, the new generation is still cataloged, and you would have to update the job. DISP=(NEW,CATLG,DELETE) would be a better choice here.

Usage & Invocation

- How can you tell with GDGBIAS setting is in use?
  - Allocation will issue a new message at the start of the job (to the job's system message data set) that indicates the GDGBIAS setting that the job will use, based on the JOB statement and the job class information:

```
IEFA111I jobname IS USING THE FOLLOWING JOB RELATED SETTINGS:
         SWA=sssss,TIOT SIZE=nnK,DSENQSHR=ddddddddd,GDGBIAS=gggg
```

    Note that this message includes several settings, not just GDGBIAS.
  - Programs can also check the GDGBIAS setting in use via a new field in the JMR.
    - The JMRGDGBIASSTEP flag will be on when GDGBIAS=STEP is in effect, and off when GDGBIAS=JOB is in effect.
    - Programs can get to the JMR via TCB->TCT->JMR.
  - The JES properties SSI function 82 returns new flag CLSG1GST in the Job Class General Information Section.
  - The $D JOBCLASS,LONG and *INQUIRY,C=class commands have also been updated.

Usage & Invocation

- Other externals:
  - For JES2, the $ADD JOBCLASS and $T JOBCLASS commands have been updated to support GDGBIAS, in addition to $D JOBCLASS.
  - For JES3, the *MODIFY,C=class command has been updated to support GDGBIAS, in addition to *INQUIRY,C=class.
  - See the Installation page for examples of job class definitions.

Usage & Invocation

- For customers that have traditionally used their own USERMOD to implement this functionality:
  - It seems that although these USERMODs accomplish similar functions, each installation seems to have their own unique implementation.
  - IBM has built its implementation of GDGBIAS=STEP based on how we have been told these USERMODs work, and not using any specific customer's version.
    - We expect that in most cases, our version should have the same results as your USERMOD. However, it is possible that there are some differences.
    - Since there may be differences in the USERMOD implementation from customer to customer, we may not be able to change our implementation to match your USERMOD's behavior. It may be necessary to change your JCL to match our implementation.

Migration & Coexistence Considerations

- No special toleration or coexistence APARs are needed for this support.
- The GDGBIAS keyword is only supported on z/OS 2.3.
  - Jobs that specify the GDGBIAS keyword must be converted, interpreted, and executed on z/OS 2.3.
    - This is true for GDGBIAS=JOB or GDGBIAS=STEP. If you need a job to run on lower releases, do not code the GDGBIAS keyword!
  - For jobs that do not have the GDGBIAS keyword specified, JES will only process jobs in a job class with GDGBIAS=STEP on z/OS 2.3 systems.
    - Jobs in a job class that uses GDGBIAS=JOB can execute on prior releases as well as z/OS 2.3 systems.

Installation

- For most customers, there are no installation steps needed.
- For customers that have a custom USERMOD to implement the alternate GDG functionality, it will probably be necessary to define or modify job classes with GDGBIAS=STEP.
- Examples of job class definition statements:
  - JES2:
    ```
    JOBCLASS(S) SWA=ABOVE,GDGBIAS=STEP,MSGLEVEL=(1,1),
    GROUP=TESTINIT
    ```
  - JES3:
    ```
    CLASS,NAME=S,SYSTEM=JES3,GROUP=JES3EXEC,
    JOURNAL=YES,GDGBIAS=STEP
    ```

Session Summary

- When using GDGBIAS=STEP, the association between relative and absolute generation numbers is resolved on a step basis rather than a job basis.
  - This is intended for customers that have custom USERMODs to provide this function today.
  - Customers that restart jobs that use GDGs may also find it to be valuable.

Appendix

- Publications:
  - SA23-1386 z/OS MVS JCL User's Guide
  - SA23-1385 z/OS MVS JCL Reference
  - SC23-6862 z/OS DFSMSdfp Checkpoint/Restart

Appendix

- Example 1:
  GDG01 has 3 existing generations (GDG01.G0001V00, GDG01.G0002V00, GDG01.G0003V00) prior to the start of the job:

```
//JOB1 JOB GDGBIAS=STEP
//STEP1    EXEC  PGM=IEFBR14
//DD1      DD    DSN=GDG01(+1),DISP=(NEW,CATLG),
//         SPACE=(TRK,1)
//DD2      DD    DSN=GDG01(-1),DISP=(OLD,KEEP)
```

- DD1 will cause a new data set to be created, GDG01.G0004V00.
- DD2 will allocate existing generation data set GDG01.G0002V00.
- Note that in this case, GDGBIAS=JOB would perform the same way, since this is a single step job. The creation of the new (+1) data set affects subsequent steps, not the current step.

Appendix

- Example 2:
  GDG01 has 3 existing generations (GDG01.G0001V00, GDG01.G0002V00, GDG01.G0003V00) prior to the start of the job:

```
//JOB1 JOB GDGBIAS=STEP
//STEP1    EXEC  PGM=IEFBR14
//DD1      DD    DSN=GDG01(+1),DISP=(NEW,CATLG),
//         SPACE=(TRK,1)
//STEP2    EXEC  PGM=IEFBR14
//DD2      DD    DSN=GDG01(-1),DISP=(OLD,KEEP)
```

- For STEP1, the (0) level is associated with GDG01.G0003V00.
- DD1 will cause a new data set to be created, GDG01.G0004V00.
- For STEP2, the (0) level is now associated with GDG01.G0004V00.
- DD2 will allocate existing generation data set GDG01.G0003V00.

Appendix

- Example 3:
  GDG01 has 3 existing generations (GDG01.G0001V00, GDG01.G0002V00, GDG01.G0003V00) prior to the start of the job:

```
//JOB1 JOB GDGBIAS=STEP
//STEP1    EXEC  PGM=IEFBR14
//DD1      DD    DSN=GDG01(+1),DISP=(NEW,PASS),
//         SPACE=(TRK,1)
//STEP2    EXEC  PGM=IEFBR14
//DD2      DD    DSN=GDG01(+1),DISP=(OLD,CATLG)
//STEP3    EXEC  PGM=IEFBR14
//DD3      DD    DSN=GDG01(0),DISP=(OLD,KEEP)
```

- DD1 will cause a new data set to be created, GDG01.G0004V00, but it is not cataloged at this time.
- Since STEP1 did not catalog the new data set, DD2 must refer to the (+1) data set to allocate the GDG01.G0004V00 created in the prior step and catalog it.
- Since STEP2 cataloged the data set, DD3 must refer to it as (0) to allocate the same GDG01.G0004V00 data set.

Appendix

- Example 4:
  GDG01 has 3 existing generations (GDG01.G0001V00, GDG01.G0002V00, GDG01.G0003V00) prior to the start of the job:

```
//JOB1 JOB GDGBIAS=STEP
//STEP1    EXEC  PGM=IEFBR14
//DD1      DD    DSN=GDG01(+1),DISP=(NEW,CATLG),
//         SPACE=(TRK,1)
//STEP2    EXEC  PGM=IRXJCL,PARM='TESTREXX'
```

  where the TESTREXX exec contains:

```
/* REXX */
allocstr = "alloc fi(dd2) da('GDG01.GDG(-1)') old"
Call BPXWDYN(allocstr)
```

- For STEP1, the (0) level is associated with GDG01.G0003V00.
- DD1 will cause a new data set to be created, GDG01.G0004V00.
- For STEP2, the (0) level is now associated with GDG01.G0004V00.
- DD2 will allocate existing generation data set GDG01.G0003V00.
- Note that this is exactly the same behavior as example 2.