

Misconceptions and Mysteries About Generation Data Groups

By Bob Perego

Misconceptions and mysteries abound about Generation Data Groups (GDGs) and Generation Data Sets (GDSs). GDSs are also referred to as “G0000V00s”, a slang term derived from the fact that the last qualifier of a GDS has the format GnnnnVnn. This article discusses some of the common misconceptions and mysteries.

Cataloging Active Generations

A common misconception is about how GDSs are cataloged. If you enter the base name.* into an ISPF 3.4 display, it will appear as though each GDS is a discrete catalog entry. For *active* GDSs this is not true. In reality, the existence of all active generations and catalog information about them are stored in a catalog type “B” record or the Base record for the GDG. This record also contains the GDG attributes, such as LIMIT and SCRATCH versus NOSCRATCH. It is not uncommon for a GDG to exceed the capacity of a type “B” record when many active generations are being retained and generations occupy many volumes. Additional space is acquired by using catalog type “J” extension records.

A part of the GDG Base catalog records called the Generation Aging Table, or GAT, establishes the relativity of generations. Even though the nnnn portion of the GnnnnVnn qualifier is used to determine the placement of an entry in the GAT when creating or recovering GDSs, it is the physical relativity of the GAT entries that determines the desired GDS when referencing a generation by a number in parenthesis after the base name.

The cataloging mechanism for GDGs therefore presents some interesting implications. For instance, if something should happen to the base catalog entry, all catalog access to the active GDSs is lost. In a copy or move scenario, the base catalog record is the only repository for the correct sequence of generations and the overall GDG attributes. Therefore, if the base entry is not also copied or moved in concert with the data, resurrecting the GDG may pose problems.

Deferred Rollin and Rolled-Off Catalog Entries

Generations are initially cataloged in Deferred Rollin status and become active at deallocation time, unless for some reason they cannot be cataloged as active generations. Generations to be kept even though they “roll off” the active list are referred to as Rolled-Off generations. Both Deferred Rollin generations and Rolled-Off generations have discrete catalog entries, often referred to as nonVSAM catalog entries.

Hence, even though displays such as ISPF 3.4 list all GDSs with names as they would be found in the VTOCs, in reality only the Deferred Rollin and Rolled-off GDSs have discrete catalog entries.

Active generations are all represented in the GDG base record.

The Automatic Wrap Mechanism

Around 1999, IBM introduced the capability for generations to wrap upon exceeding generation number 9999. Understanding the internals of this mechanism is an academic exercise, unless something goes astray or the usage of a GDG doesn't conform to a rule IBM introduced.

When wrapping around from a high-numbered generation, such as generation 9999, the GDG processing logic must have some way to know that generation 0001 comes after generation 9999 rather than before it. Given the GDS limit of 255, someone only accustomed to incrementing generations by ones might think that the order of generations 9999 and 0001 is obvious. However, because there is no rule that generations must increment by ones, the wrap mechanism must take this into account.

The solution is a "wrap" flag set for lower-numbered generations that are created as a result of wrapping past generation 9999. This signifies that the lower-numbered generations come after the higher-numbered generations. However, at some point the wrap flags must be reset. This occurs when the last of the higher-numbered generations is rolled off or if the GDG is emptied.

Another wrap mechanism issue that IBM had to consider is ordering generations when absolute generation numbers are used to create GDSs, which is common when you are restoring GDGs. Although schemes involving large gaps in generation numbers are not common, IBM had to consider this in the solution. Absurd as the example may seem, if you are restoring generations 2000 and 7000, which would be generation 0 and which would be generation -1?

To resolve this, IBM created a rule for predictable wrapping, most easily expressed as "the numerical range of generations must not exceed 1000". For the majority of GDGs this rule is reasonable, given that most generations increment by ones and the limit of active generations is 255. This limit can be a concern for GDGs where a scheme is employed where generation numbers do not necessarily increment by ones. Knowing this rule demystifies what might seem like strange GDG behavior.

Why Do Some Messages Refer to Limit 10,999?

Messages that refer to a limit of 10,999 are disconcerting given that the highest generation number is 9999. This message is disconcerting given that the highest generation number is 9999. Someone had the thought that when wrapping past generation 9999, the wrap bit mechanism might be best understood by thinking of the new lower-numbered generations as 10,nnn until the wrap flags are turned off. The message "Cataloging G1000Vnn will cause GDG to exceed limit of 10,999" means that an attempt is being made to create generation 1000 before the last of higher-numbered generations causing the wrap have been rolled off. Again, considering that a numerical range of 1000 for active generations is liberal for most scenarios, this error will most often occur when something goes astray.

Backing Up and Restoring GDSs

The easiest way to avoid problems is to back up the base catalog record and all active generations in a single execution of a utility. For example, ABARS will capture the base record along with generations selected for backup. When restoring, ABARS will also restore the generations in the relativity sequence, hence avoiding any issues with keeping correct relativity and

ensuring that the GDG attributes are current.

Restoring generations out of sequence can cause problems. Restoring more generations than the LIMIT can also yield undesirable results.

GDG processing attempts to use the GnnnnVnn qualifier of data set names to set the relativity in the GAT to what might be expected. For instance, even though generation `basename.G0002V00` might be restored first, followed by `basename.G0001V00`, generation `G0002V00` will end up in the GAT as the current generation and `G0001V00` as the -1 generation.

However, this may not hold true when active wrapped generations are restored. For instance, if you start with an empty GDG base and restore `basename.G0001V00` first, followed by `basename.G9999V00`, GDS 9999 will end up being the current generation and GDS 0001 the -1 generation, likely the reverse of the relativity for the generations in the original GDG .

When you are restoring more generations than the LIMIT, be aware of another phenomenon: if the GDG is full, i.e., the active generations equals the LIMIT, and a lower-numbered GDS is added, the lower-numbered GDS will replace the oldest generation in the GAT. This replacement can easily happen if generations that are no longer active are restored.

The safest backup/restore methodology is to back up and restore the populated base record and to back up and restore the generations in order from oldest to newest. This will ensure that GDSs including a wrap are restored correctly and that the base attributes are current.

Generation 9000, Another “Magic” Number

Creating GDS 8999 followed by creating GDS 0001 will result in GDS 8999 being the current generation, or (0). However, creating GDS 9000 followed by creating GDS 0001 will result in GDS 0001 being the current generation, or (0).

Consider again that generations need not be sequenced by ones. A compromise had to be made between trying to restore generations based on the GDS number and honoring the fact that GDSs may be wrapped. Despite the fact that a gap this large in GDS numbers is unusual, the authors of the code had to choose some generation number to decide the GAT placement of a generation being added.

Learning More Through Experimentation with Catalog RecoveryPlus (CR+)

The CR+ ZAP command will display the contents of a GDG base record. Select the ISPF ZAP option and enter the GDG base name in the “Key” field. BCS name may be left blank. For online viewing, enter an “S” next to “View/Create VER & REP”. Enter “F” for Print Format.

Cell type X'05' is the GAT. The first fields of the GAT contain flags and values for the base attributes. Following are repeating groups of fields for each generation possible, determined by the LIMIT. For easy identification, the CR+ ZAP formatting separates the fields with a comment line showing the relativity as it would be specified in JCL, i.e., 0, -1., etc. Another comment makes the wrap bit being set quite obvious. When in wrap mode, the generation is also shown with its “pseudo” number, i.e., 10,nnn.

Because the ZAP display has a REFresh option, experiment by leaving one panel set with the ZAP display. From another panel or session, experiment deleting and allocating GDSs, then enter the REF command to see how changes have affected the GAT. Use a **test GDG** for experimentation, so that you can simulate many of the backup/restore scenarios by simply allocating

Misconceptions and Mysteries About Generation Data Groups

and deleting GDSs via ISPF panel 3.2.

You will note that in the GAT, as well as in other cell types of the GDG base record, generations are referenced only by the nnnn portion of GnnnnVnn. To bring this discussion full circle to the beginning, consider how Panel 3.4 displays active generations when ISPF displays each GDS name in full, i.e., basename.GnnnnVnn, even though in reality the name list is being constructed using the basename and each active generation number found in the catalog base record.

Conclusion

By understanding how GDGs work, you can avoid potential problems. With **CR+**, you can experiment and see how GDGs would be affected by various backup and restore scenarios.

Please contact Mainstar Technical Support if you have more questions or problems with GDGs.

Bob Perego – Director of Software Development

Bob has 40 years of experience and a deep understanding of IBM mainframe operating systems. He manages the design and the development of Mainstar's software products.

Mainstar is a registered trademark and Catalog RecoveryPlus is a trademark of Mainstar Software Corporation.

IBM is a registered trademark and Redbooks, DFSMSdss, and DFSMSHsm are trademarks of the International Business Machines Corporation.