PDSE Version 2: Member Generations Practical User Applications

Speaker: Thomas Reed /IBM Corporation

SHARE Pittsburgh 2014

Session:16127

#SHAREorg

Agenda

- PDSE Member Generations Basics
- Working with Member Generations
- ISPF Support
- Member Generations Macro Support
- Member Generations REXX tutorial
  - Data Set Information
  - Listing Members
  - Discovering Generations
  - Recovering Generations

What is a PDSE?

- PDSE: Partitioned Data Set Extended
- A PDSE is a collection of directory and data pages
- At V2R1 there are 2 data set formats V1 and V2 PDSEs
- PDSE server consists of one or two address spaces (SMSPDSE and SMSPDSE1)
- The SMSPDSE(1) address spaces serve client access requests for PDSE data sets
- Under the hood SMSPDSE(1) also manages PDSE serialization and buffering

PDSE Versions

- At V2R1 there are 2 data set formats V1 and V2 PDSEs
- The version 1 format is the historic PDSE format
- The version 2 format is a revision of the PDSE format
  - Brings better performance and efficiency
  - Reduces CPU and Storage utilization
  - **Supports PDSE member generations**
- Version 2 data sets use the same serialization and buffering subsystems as version 1
- The IMF/BMF performance enhancements at V2R1 apply to BOTH V1 and V2 datasets

PDSE Version 2: Member Generations

- Implemented via DFSMS APAR OA42358
  - ISPF Support via APARs OA42247 and OA42248
- Exclusive to the V2 PDSE Format
- PDSE Data sets can now retain multiple generations of members
- Applies to BOTH Data Members and Program Objects
- Retains generations up to the data set/system limit

PDSE Version 2: Member Generations

Terminology

- Generation (GEN)
  - A prior copy of a member
- Primary Generation
  - The current member
  - Absolute and Relative 0
- Generation Numbering
  - Absolute: GEN(n), GEN(n-1), GEN(n-2)….
  - Relative: GEN(-1), GEN(-2),….,GEN(-MAXGENS)
    - n being the nth generation created

PDSE Member Generations: The Basics

- FIFO (First In, First Out) structure
  - Oldest generation is permanently deleted if it's over the generation limit
  - Old generations generally behave just like primary members
  - Aliases are retained for previous generations and can be recovered*

* When STOW RECOVERG is used

PDSE Member Generations: The Basics

- Generations are uniquely numbered
  - They can be referenced either by their **Absolute** or **Relative** generation
  - The Primary Member is always 0, both relative and absolute
  - Greatest number indicates the newest generation

PDSE Member Generations: The Basics

- Example: MAXGENS = 4 after 10 generations

ABS	0	1	2	3	4	5	6	7	8	9	10
	PRI	-	-	-	-	-	-	Gen	Gen	Gen	Gen
REL	0	-	-	-	-	-	-	-4	-3	-2	-1

- Note that the newest generation ALWAYS has the greatest value

PDSE Member Generations: The Basics

Usage Considerations

- Allow extra space for each generation
- Each generation retains the entire member
- MAXGENS_LIMIT in IGDSMSxx is the System limit
- MAXGENS_LIMIT can be set dynamically via SET SMS=xx
  - Cannot be set dynamically with SETSMS
- MAXGENS_LIMIT upper limit is set at 2 billion

PDSE Member Generations: Working with Generations

Enabling Member Generations

1. Ensure that the requisite APARs are applied
2. MAXGENS_LIMIT needs to be set >0 in your IGDSMSxx
3. Allocate a V2 PDSE dataset with greater than 0 generations (must be <= MAXGENS_LIMIT)

PDSE Member Generations: Coexistence

- Down level systems will tolerate V2 PDSE's with Generations and be able to open for INPUT of OUTPUT
- Down level systems will not be able to create or manipulate generations
- DFSMSdss support is identical to 2.1
  - DSS Copy will retain generations with OA43729 or Concurrent Copy
  - Logical or Physical DUMP and RESTORE retains generations

PDSE Member Generations: Working with Generations

Allocating a PDSE with Generations Enabled via JCL!

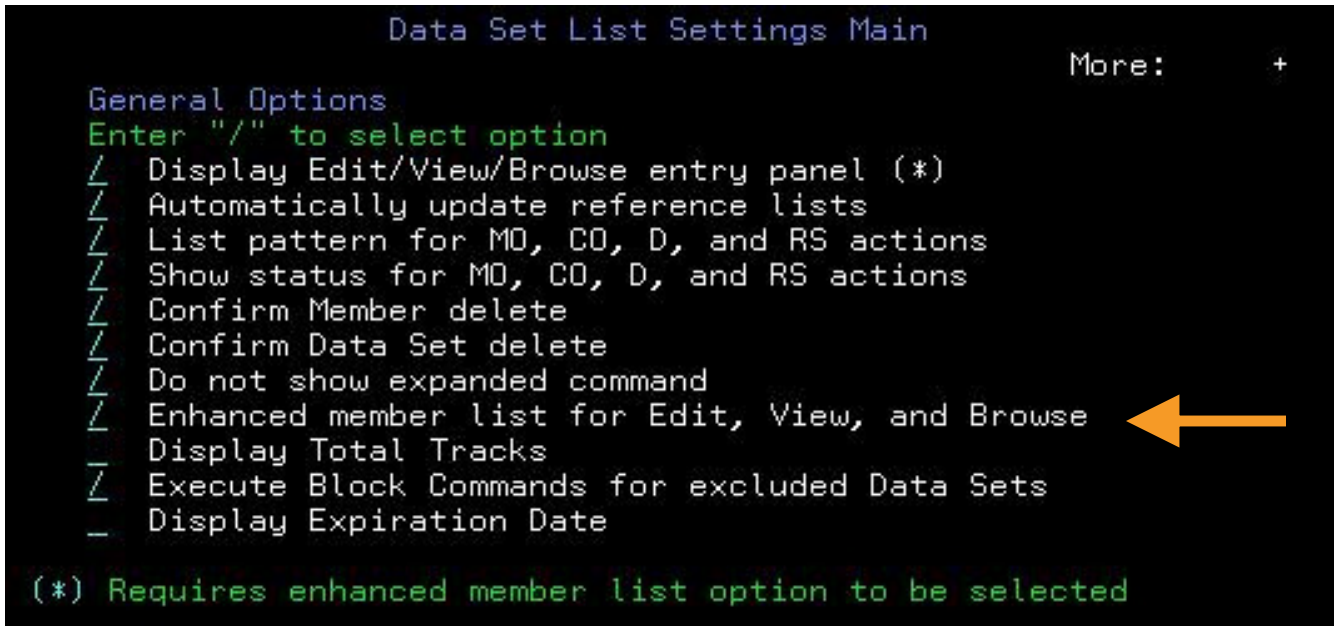
```
//ALLOC     EXEC PGM=IEFBR14
//PDSE2     DD DSN=TREED.PDSE.GENS,
// DSNTYPE=(LIBRARY,2),MAXGENS=10,
// RECFM=FB,LRECL=80,
// UNIT=SYSALLDA,SPACE=(CYL,(1,1,1)),
// DISP=(,CATLG,DELETE)
```

- Note that LIBRARY,2 specifies a V2 data set
- MAXGENS must be <= the system MAXGENS_LIMIT

PDSE Member Generations: ISPF Support

Panels

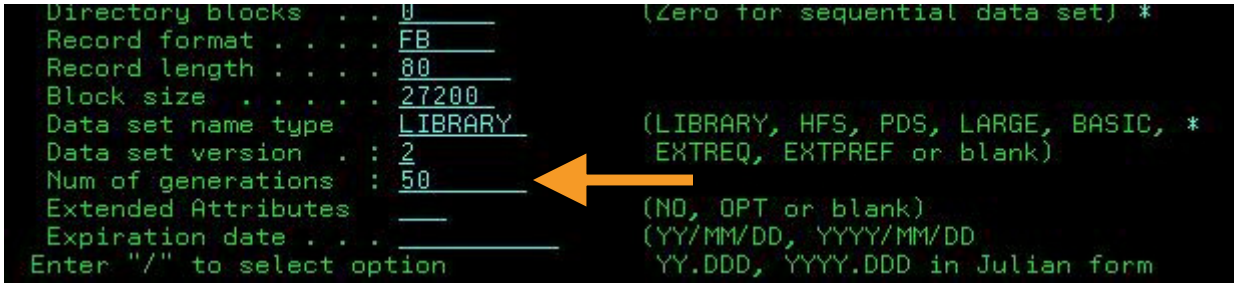
- ISPF now has generations support
- Enhanced member list option must be selected



PDSE Member Generations: ISPF Support

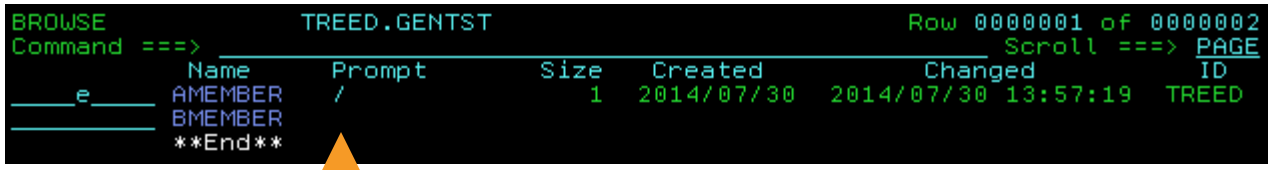
Allocation

- Allocates like any other PDSE
- MAXGENS must be >0
- Be sure you're using version 2!

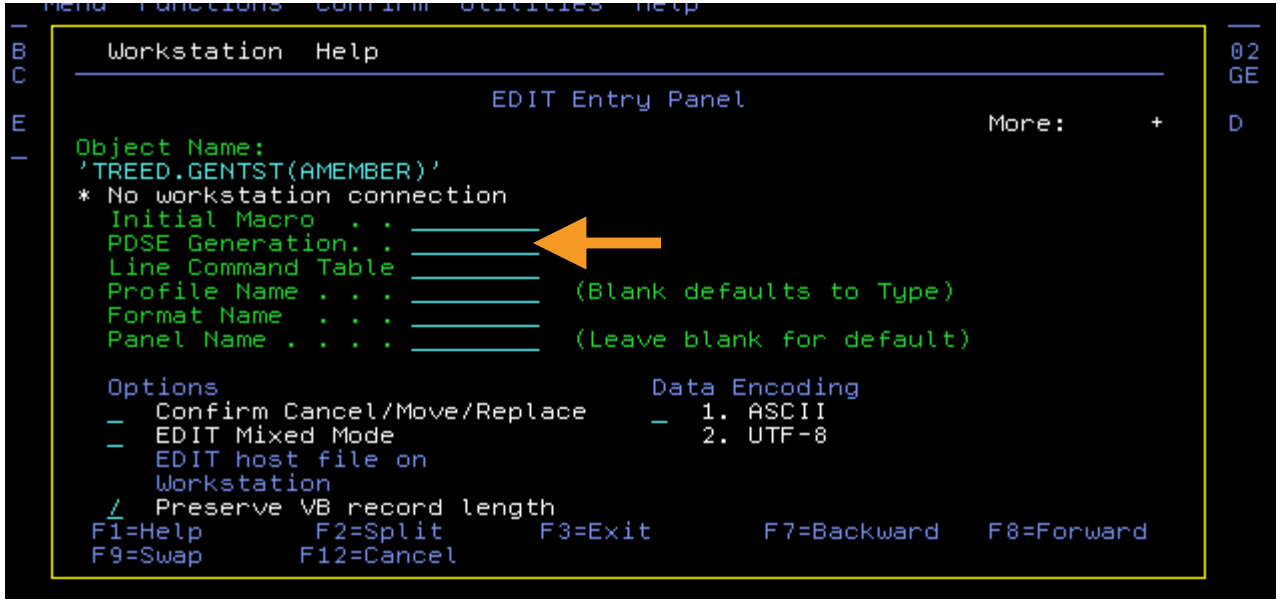


PDSE Member Generations: ISPF Support

- Accessing generations through 3.4



Don't forget the '/'



PDSE Member Generations: ISPF Support

Restrictions

- ENQUEUEing on one generation applies to all generations of that member
  - This is not a PDSE serialization restriction
  - The native API's allow for editing of multiple generations of the same member
- ISPF Options 1 and 2 do not support a GEN parameter
- ISPF 3.1 and 3.4 do support a GEN parameter

PDSE Member Generations: Working with Generations

Creating a Generation

- 2 requirements
  - (LIBRARY,2)
  - MAXGENS > 0
- New generations are automatically created on replace or delete of a member
- Update in place will not create a new generation
- Generation creation is atomic

PDSE Member Generations: ISPF Support

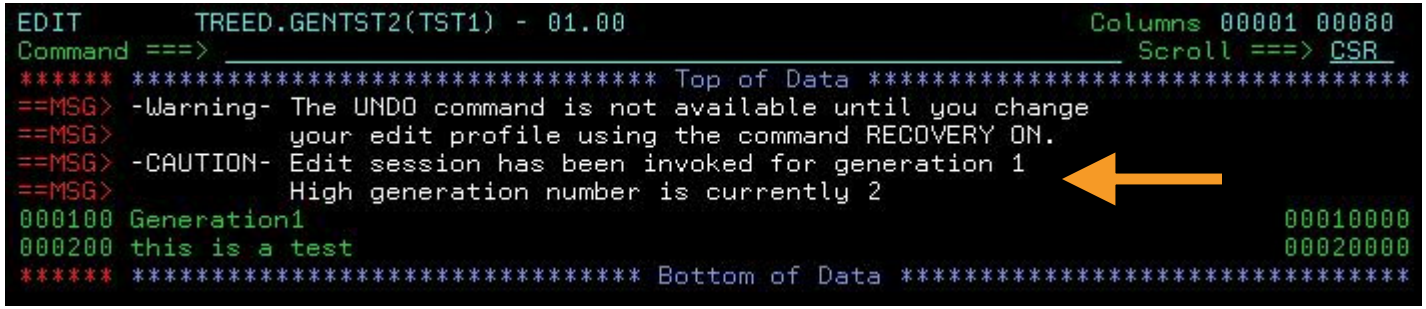
Editing

- Editing the current member (GEN 0) results in a new generation being created
- Editing prior generations does NOT result in a new member
- Supports referencing generations by either absolute or relative generation number
- Deleting a member in ISPF deletes all generations
  - This is an ISPF implementation feature
  - TSO DELETE pdse(member) deletes only the primary

PDSE Member Generations: ISPF Support

Editing Cont'd

- Generation creation behavior can be forced
  - SAVE NEWGEN – Creates a new generation
  - SAVE NOGEN – Does not create a new generation
- Edit will tell you which absolute generation you are working with



PDSE Member Generations: Working with Generations

Reading Old Generations

- FIND macro will allow programs to connect to old generations
- Conventional READ and CHECK macros still apply
- Old generations cannot be accessed via JCL or dynamic allocation

PDSE Member Generations: Working with Generations

Deleting Old Generations

- Each generation must be deleted separately
- Deleted generations can be replaced by using STOW RG
- ISPF member delete will delete all generations

PDSE Member Generations: Working with Generations

Recovering Old Generations

- Read an old generation and then write it to either the same or a different member name
  - The old generation will become the current generation
  - Note: This method will not restore aliases
- Use the RECOVERG option for the STOW macro
  - The old generation becomes the current generation of the member of the same name
  - Note: Aliases ARE recovered by this method

PDSE Member Generations: Working with Generations

Backup Considerations

- IEBCOPY and IDCAMS REPRO
  - Only copy the current generation of each member
  - All old generations are lost
- DFSMSdss
  - Physical or Logical dump and restore retain all old generations
  - This includes HSM backup

PDSE Member Generations: DESERV Macros

FUNC=GET_G (AKA Get Generation)

- Returns information for the selected generation
- Returns the same information as GET plus the relative and absolute generation numbers
- A dummy entry is returned if the selected generation does not exist
- Does not support CONNECT

PDSE Member Generations: DESERV Macros

FUNC=GET_G

,AREA=(buffer_area, buffer_area_size)

,DCB=data_control_block

,NAME_LIST=(generationname,1)

[,MF={(E,parmlist_name[,NOCHECK|COMPLETE])|S}]

[,RETCODE=return_code]

[,RSNCODE=reason_code]

PDSE Member Generations: DESERV Macros

FUNC=GET_ALL_G (AKA Get All Generations)

- Returns information for the selected generation for all members
- Returns the same information as GET_ALL plus the relative and absolute generation numbers
- A dummy entry is returned if the selected generation does not exist for a member
- Does not support all the same options as GET_ALL

PDSE Member Generations: DESERV Macros

FUNC=GET_ALL_G

,AREA=(buffer_area, buffer_area_size)

,DCB=data_control_block

,NAME_LIST=(generationname,1)

[,MF={(E,parmlist_name[,NOCHECK|COMPLETE])|S}]

[,RETCODE=return_code]

[,RSNCODE=reason_code]

PDSE Member Generations: STOW Macro

DG (Delete Generation)

- Deletes an existing generation
- Takes a member name and generation number
- Leaves a gap in the generation list
- If issued with a generation of 0, deletes the member without creating a generation

PDSE Member Generations: STOW Macro

RG (Replace Generation)

- Replaces an existing generation
- Adds a generation if replacing a gap in the generation list

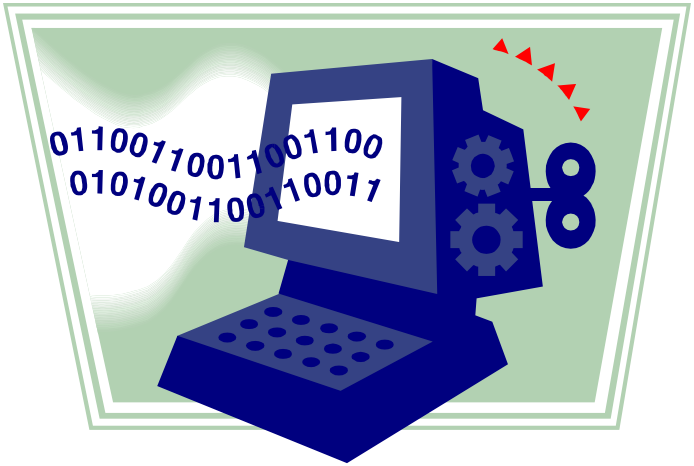
PDSE Member Generations: STOW Macro

RECOVERG (Recover Generation)

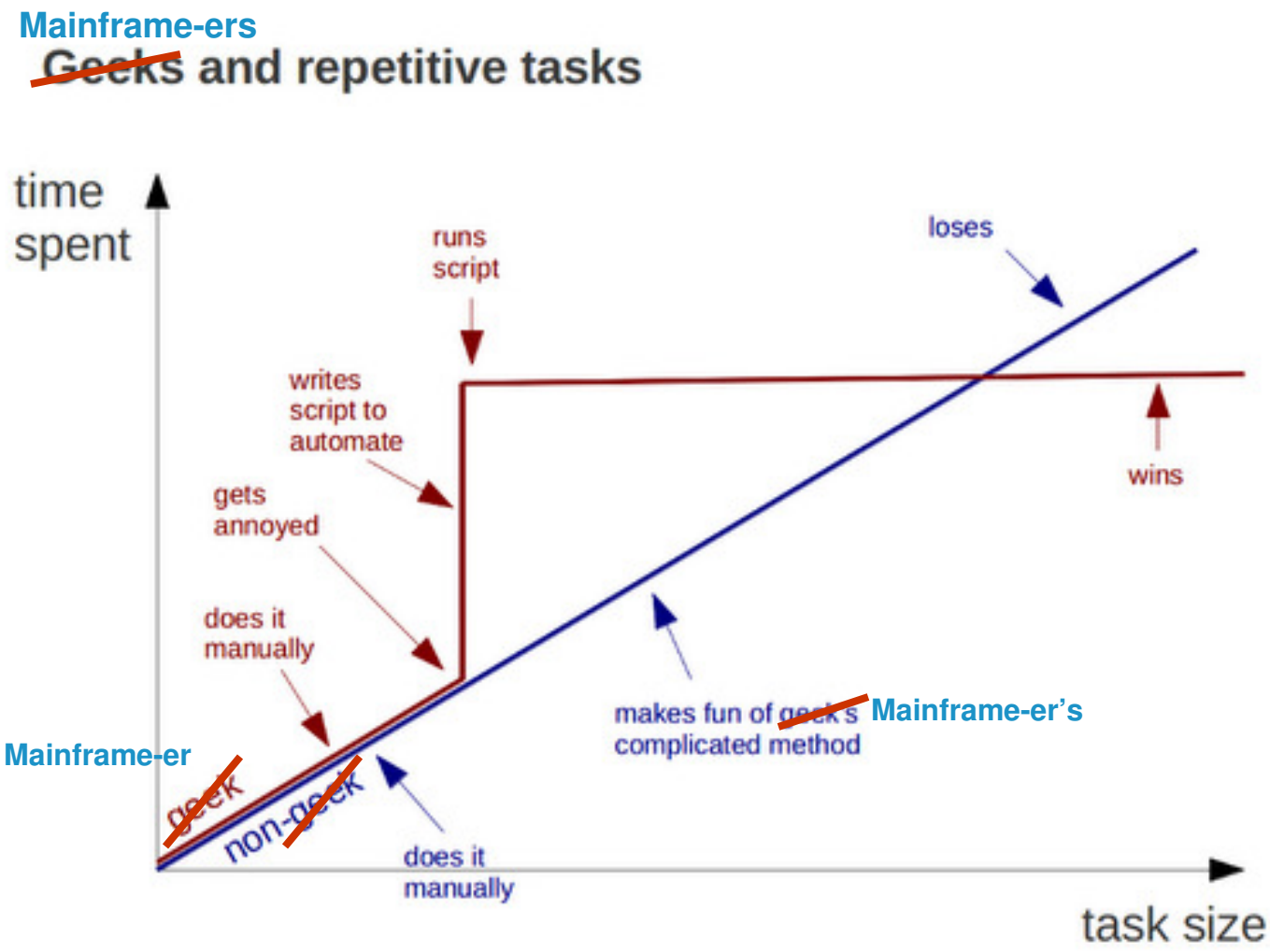
- Recovers an existing generation
- Removes the selected generation from the generation list and makes it the primary member
- Creates a new generation in the replace process from the former primary member

PDSE Member Generations: What do we do with it?!

- Manually entering generation numbers in ISPF
  - Time consuming
  - No generation list
- We can get at these same interfaces programmatically!
  - Examples will be in REXX



PDSE Member Generations: What do we do with it?!



PDSE Member Generations: What do we do with it?!

- PDF API Changes for Generations
  - DSINFO
    - ZDSDSNV = The version of the PDSE
    - ZDSNGEN = The number of generations specified (MAXGENS) on allocation of the PDSE
  - LMDSLIST
    - ZDLDSNV = The version of the PDSE
    - ZDLNGEN = The number of generations specified (MAXGENS) on allocation of the PDSE
  - EDIT VIEW BROWSE support the GEN parameter
    - GEN(n) = Either relative or absolute generation
  - EDIT SAVE supports NOGEN and NEWGEN

PDSE Member Generations: Code Disclaimer

- This code is UNSUPPORTED and is intended only to provide usage examples
- These examples are provided as is with no guarantees as to their correctness of effectiveness
- IBM is not responsible for damages or any other problems incurred through the use of these examples

PDSE Member Generations: Generations in Code

- Getting more information about your PDSE
  - DSINFO or LMDSLIST
  - Both call the same FAMS interface underneath

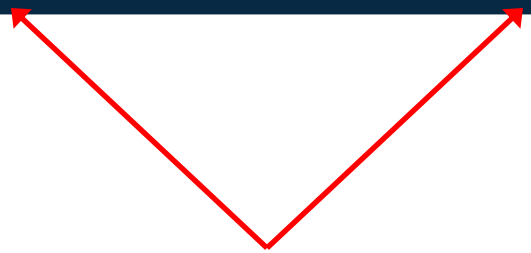
- This lets us determine the VERSION and MAXGENS of the PDSE
  - No point in trying to manipulate generations in a V1 PDSE
  - If MAXGENS is 0 then it won't work either

PDSE Member Generations: Generations in Code

- Finding out about your PDSE

```
/*********************************************************************/
/* Show what type of data set it is                                  */
/*********************************************************************/
ADDRESS ISPEXEC 'DSINFO DATASET('TargetDataSet')'

SAY 'VERSION:'ZDSDSNV' Generations:'ZDSNGEN
```



- DSINFO sets many variables
- New for 2.1
  - ZDSDSNV = The version of the PDSE
  - ZDSNGEN = The number of generations specified (MAXGENS) on allocation of the PDSE

PDSE Member Generations: Generations in Code

- How do I list the members?
  - LISTDS with the MEMBERS option
  - Returns Data Set information and Member List
    - We can discard the first 6 lines
    - We only need the member list

- This only gets us the list of PRIMARY members in the PDSE
  - This doesn't tell us anything about each member's generations, if they have any

PDSE Member Generations: Generations in Code

- Listing Members

```
/*********************************************************************/
/* List the members in the data set                                  */
/*********************************************************************/
call outtrap "LIST."
"LISTDS "TargetDataSet" MEMBERS"
call outtrap "OFF"
DO i = 7 to list.0 /* 1 to 6 contains info about the DS*/
  member = strip(list.i)
  SAY 'Member Name: 'member
END
```

- "TargetDataSet“ is our DSN
- This will work for V1 and V2 PDSE’s

PDSE Member Generations: Generations in Code

- How do I list generations for a member?
  - First we start with a PRIMARY member name
  - EDIT now supports a GEN(n) parameter
    - Takes either absolute or relative generation
  - EDIT will report a RC=10 if a generation does not exist

- We don't actually want to EDIT the generation, only see if it exists.

PDSE Member Generations: Generations in Code

- How to EDIT without EDITING
  - The NOED macro
  - Also the sound of one hand clapping

```
/*REXX MACRO PROGRAM*/
"ISREDIT MACRO PROCESS"
"ISREDIT CANCEL"
ADDRESS 'ISPEXEC'
RETURN
```

- The macro simply cancels the EDIT session
  - Prevents updates to the generation
  - Prevents the EDIT dialog from showing on screen
  - CANCEL causes a RC=4

PDSE Member Generations: Generations in Code

- Listing Absolute Generations

```
/*********************************************************************/
/* See what absolute generations exist                               */
/*********************************************************************/
ADDRESS ISPEXEC 'LMINIT DATAID(DATAIDV) DATASET('TargetDataSet')'
DO j=0 to 50
  ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
  ADDRESS ISPEXEC 'EDIT DATAID('DATAIDV') MEMBER('member') GEN('j') MACRO(NOED)'
   IF RC<=4 Then
     SAY 'Generation 'j' Exists.'
 End
```

Check the EDIT RC

- <=4, the gen exists
- >4, the gen doesn't exist or we hit an error

Generation Parm

The EDIT Macro

PDSE Member Generations: Generations in Code

- Listing Relative Generations

```
/*********************************************************************/
/* See what relative generations exist                               */
/*********************************************************************/
ADDRESS ISPEXEC 'LMINIT DATAID(DATAIDV) DATASET('TargetDataSet')'
DO j=0 to ZDSNGEN
  ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
  ADDRESS ISPEXEC 'EDIT DATAID('DATAIDV') MEMBER('member') GEN('j * -1') MACRO(NOED)'
   IF RC<=4 Then
     SAY 'Generation 'j * -1' Exists.'
 End
```

We can have up to ZDSNGEN generations per member (MAXGENS)

Generation Parm

The EDIT Macro

PDSE Member Generations: Generations in Code

- Now we can:
  - Get Version and MAXGENS
  - List PRIMARY members in the PDSE
  - Determine which generations exist for each member
- This tells us useful information that we didn't know about the V2 dataset before
- We're still not manipulating generations though!

PDSE Member Generations: Generations in Code

- How do I replace the primary with a previous generation?
  - Very similar to determining if a generation exists
  - Relative generations are far easier to work with
    - Restoring GEN(-1) for example
    - No need to reference absolute generation value

- This time we will actually EDIT the generation
  - Simply use SAVE NEWGEN to replace the primary

PDSE Member Generations: Generations in Code

- How to EDIT and create a new primary
  - The SAVENEWG macro

```
/*REXX MACRO PROGRAM*/
"ISREDIT MACRO PROCESS"
"ISREDIT SAVE NEWGEN"
"ISREDIT END"
ADDRESS 'ISPEXEC'
RETURN
```

- The macro opens the generation in EDIT
  - Simply SAVEs the open generation as the primary
  - Uses SAVE NEWGEN to force the creation of a new generation
  - Returns RC=0 on success

PDSE Member Generations: Generations in Code

- Restoring Old Generations

```
/*********************************************************************/
/* Restore generation -1                                             */
/*********************************************************************/
ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
ADDRESS ISPEXEC 'EDIT DATAID('DATAIDV') MEMBER('member') GEN(-1) MACRO(SAVENEWG)'
IF RC>0 Then
  SAY 'No Generation to Restore'
ELSE
  SAY 'Restored Latest Generation'
```

We may not have a generation to restore

Generation parm using relative referencing

The EDIT Macro

PDSE Member Generations: Generations in Code

- Roll back an entire PDSE 1 generation

```
/*********************************************************************/
/* List the members in the data set                                  */
/*********************************************************************/
call outtrap "LIST."
"listds "TargetDataSet" members"
call outtrap "OFF"
DO i = 7 to list.0 /* 1 to 6 contains info abt the DS*/
  member = strip(list.i)
  SAY 'Member Name: 'member
  /*******************************************************************/
  /* See what absolute generations exist                             */
  /*******************************************************************/
  ADDRESS ISPEXEC 'LMINIT DATAID(DATAIDV) DATASET('TargetDataSet')'
  DO j=0 to 50
    ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
    ADDRESS ISPEXEC 'EDIT DATAID('DATAIDV') MEMBER('member') GEN('j') MACRO(NOED)'
     IF RC<=4 Then
       SAY 'Generation 'j' Exists.'
   End
  /*******************************************************************/
  /* Restore generation -1                                           */
  /*******************************************************************/
  ADDRESS ISPEXEC 'CONTROL ERRORS RETURN'
  ADDRESS ISPEXEC 'EDIT DATAID('DATAIDV') MEMBER('member') GEN(-1) MACRO(SAVENEWG)'
  IF RC>0 Then
    SAY 'No Generation to Restore'
  ELSE
    SAY 'Restored Latest Generation'
END
```

PDSE Member Generations: Generations in Code

- Example Output
  - Note generations are listed in absolute referencing
  - Generation rolled back using relative referencing

```
VERSION:2 Generations:       4

Member Name: AMEMBER
Generation 0 Exists.
Generation 6 Exists.
Generation 7 Exists.
Generation 8 Exists.
Generation 9 Exists.

Restored Latest Generation

Member Name: BMEMBER
Generation 0 Exists.

No Generation to Restore
```

Questions? Comments?

Please Fill Out the Survey!

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