The PDS to PDSE conversion: A Totally Expected Journey

Speaker: Thomas Reed /IBM Corporation

SHARE Orlando 2015

Session:

#SHAREorg

Agenda

- Why are we converting?
- PDSE History
- What are PDS's being used for now?
- Getting ready for PDSE's
- The conversion process and gotchas

Why convert anyways?

PDSE > PDS

PDS

- Member space non-reusable (gas)
- 65535 track limit
- Fixed directory size
- Alphabetical directory
- Limited sharing

PDSE

- All space is reusable
- Unlimited tracks (1 vol)
- Flexible directory size
- B-Trees!
- Multi-system member level sharing

COBOL V5

- COBOL V5 executables
  - No longer Load Modules
  - Now Program Objects
- Load Modules have many limitations
  - There were limits to how much BINDER can do to work around those limitations
  - A change of format was needed

COBOL V5

- COBOL can utilize the improvements in Program Objects
  - COBOL has been able to since 2001!
  - PO can have text size of 1GB (vs 16MB for LM)
  - Long program names
  - Object Oriented COBOL
  - DLLs using the Binder
  - PO's can be page mapped to 4K pages for performance

And now for some History

The History of PDSE

- Introduced in MVS/DFP 3.2 ~25 years ago
- DFSMS 1.1
  - Extended sharing
  - Program Object support
- z/OS DFSMS 1.3
  - SMSPDSE address space
- z/OS DFSMS 1.6
  - SMSPDSE1 restartable address space
- z/OS DFSMS 2.1
  - PDSE V2 Format
  - PDSE Member Generations

Program Object History

- DFSMS 1.1
  - Introduced the Binder and Loader
  - Replaced the Linkage Editor and Program Fetch
- Further improvements were made in DFSMS 1.3 and 1.4

- Converting to Program Objects has been an option for ~15 years
- Advantages over Load Modules are clear
  - Programs will continue to want to take advantage of these features in the future
  - Performance advantages are substantial

Why stick with PDS?

- PDS is simple
  - No settings
  - Directory is simple
  - Members are simple
  - Usage is fairly transparent
  - Sharing is simple
  - Lightweight resource needs

- These are the same properties that resulted in the development of PDSE
- Simplicity is great until it becomes restrictive

Taking Stock

PDS Process Questions

- Are we sharing PDS's between sysplexes?
  - Do we need to be?
  - Is this a technical or process solution?
- PDS hygiene
  - Are we mixing load modules and data?
- PDS Performance
  - What are my caching arrangements? LLA/VLF?
- What needs converting?
  - Do I have a good accounting of the libraries that I need?

PDSE Process Questions

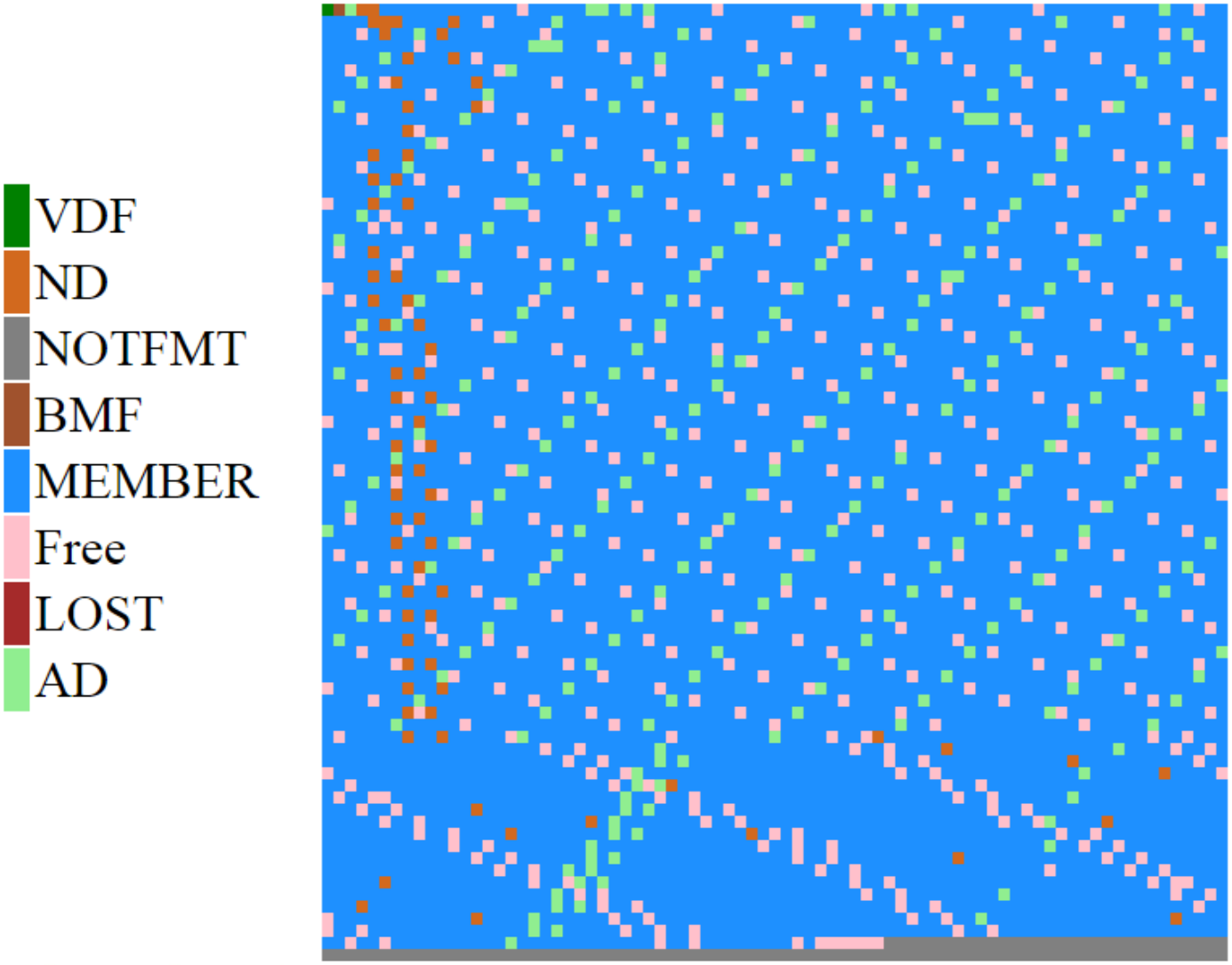
- Am I using PDSE's (Yes)?
  - What libraries are PDSE's?
  - Are they working the way I want?
- Do I have any outstanding PDSE issues?
- What are my PDSE settings?
  - Sharing Type
  - Restartable Address space
  - Buffer Beyond Close
  - Hiperspace Caching
  - V1 or V2?

PDSE Basics

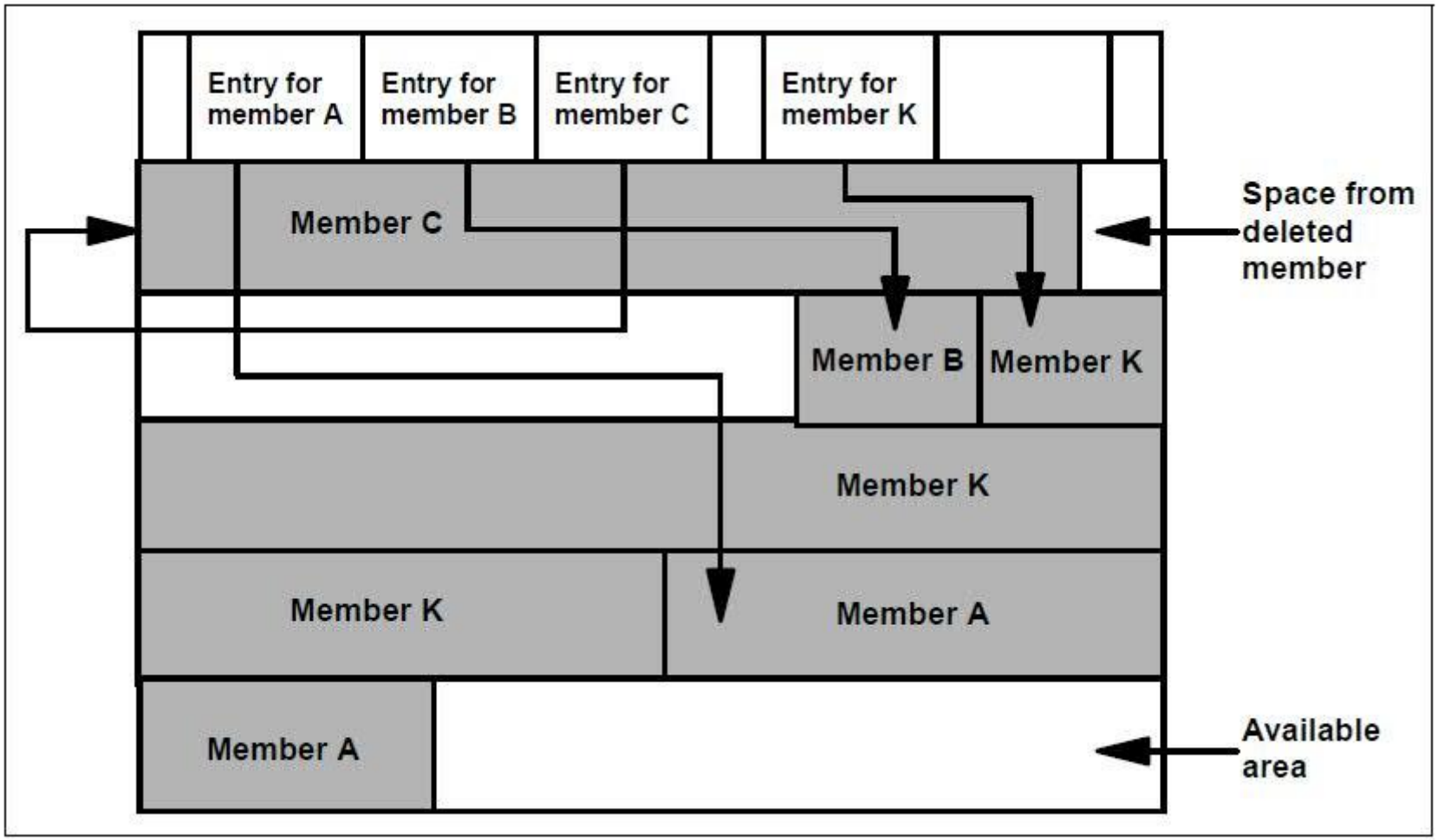
What is a PDSE?

- PDSE: Partitioned Data Set Extended
- A PDSE is a homogenous collection of directory and data pages
- 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

What Does a PDSE Look Like?



What Does a PDS Look Like?



Decision Time

PDSE Configuration

- How can I configure my PDSE environment to aid the transition?

- We have 2 options
  - Attempt to replicate our PDS environment with PDSE's
  - Attempt to leverage the PDSE's enhanced capabilities

Decision: Sharing

The Big Question

- Are we sharing PDS's outside the SYSPLEX?
  - Do we need to be?
  - Can this process be changed?

- The answers to these questions will determine the most important parts of the PDSE configuration
  - Sharing mode
  - Recoverability

PDSE Sharing Basics

Important Terminology:

- Two sharing modes, NORMAL and EXTENDED
  - NORMAL is the default mode
  - EXTENDED is preferred in the SYSPLEX environment
- GRSPLEX Scope: A set of systems connected by only GRS
- SYSPLEX Scope: A set of systems connected by both XCF and GRS

*See the Appendix for *NEW* sharing cheat sheets

EXTENDED Sharing Mode: Basics

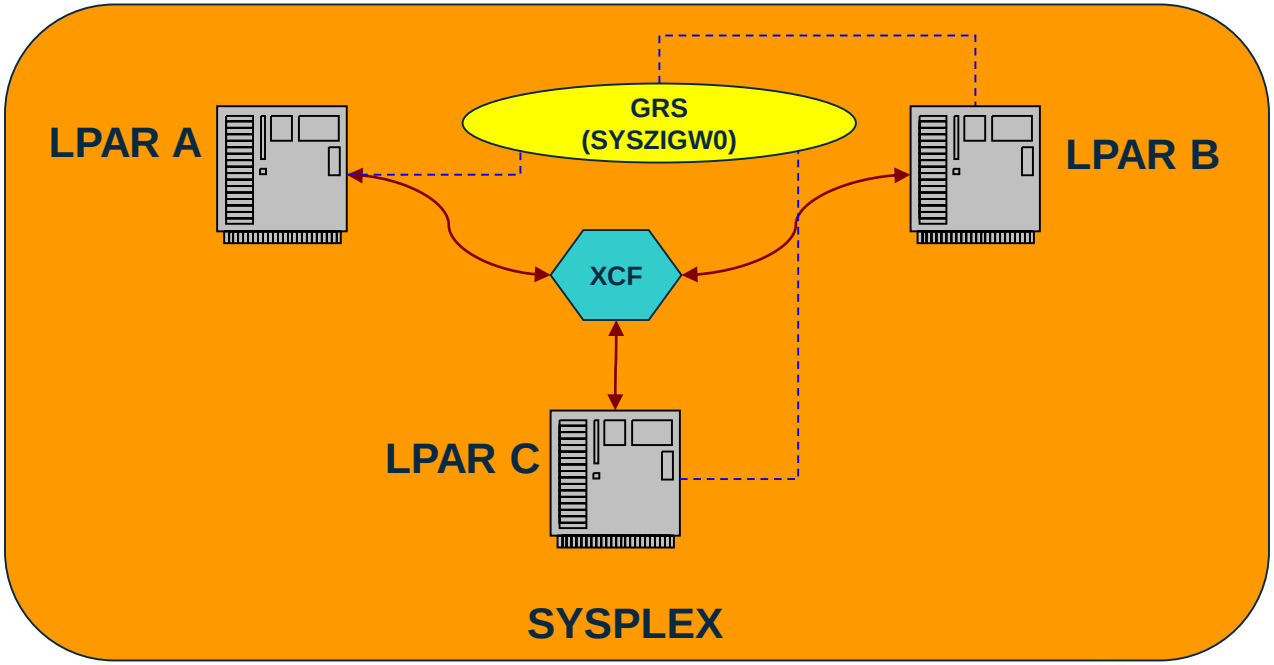
- The newest and preferred sharing mode
- Provides the ability to share at the member level between systems
- Can be implemented with one or both address spaces active

EXTENDED Sharing Mode: Startup

- `PDSESHARING(EXTENDED)` specified in IGDSMSxx member
- The SYSPLEX sharing mode is determined by the first PDSE address space to start within the GRSPLEX
- Mixed sharing modes are not supported
- IPL is recommended to start EXTENDED sharing
  - Starting with the ACTIVATE command is possible
  - ACTIVATE command start may cause PDSE problems
  - See Appendix for ACTIVATE command

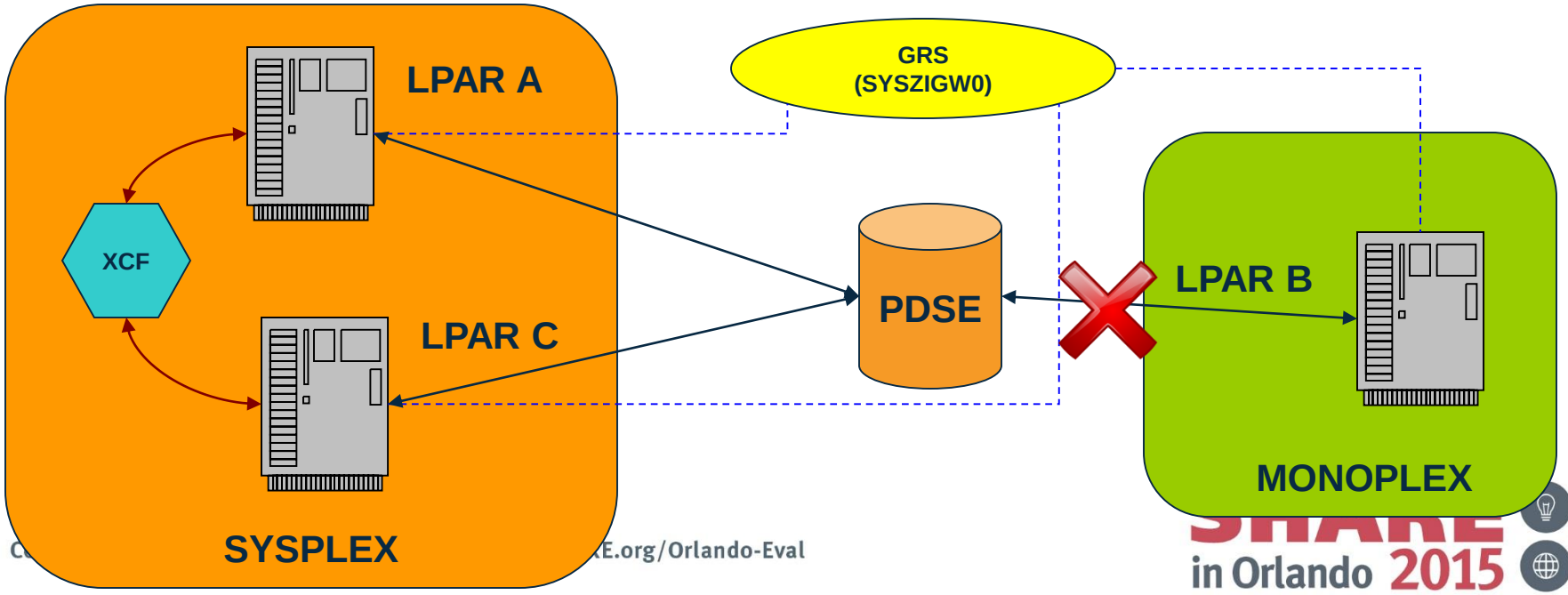
EXTENDED Sharing Mode: Sharing Requirements

- EXTENDED sharing is strictly limited to systems within the same SYSPLEX
- Participating systems must belong to the same GRSPLEX AND XCFPLEX



Improper PDSE sharing: What is it?

- Sharing a PDSE data set outside of a single XCFPLEX while running PDSE sharing EXTENDED
- Also known as sharing outside of the SYSPLEX
- Key point: PDSE sharing EXTENDED requires both GRS and XCF to mediate serialization of data sets



Improper PDSE sharing: Why is it bad?

- Improper sharing can allow for unserialized access to PDSE data sets
  - There is no warning that a data set has been accessed in an unserialized manner
  - The results are unpredictable but may include:
    - Invalid index data in-core
    - Corrupt index data on DASD
    - Corrupt member data
    - Mismatched extent information
    - Nothing at all

Improper PDSE sharing: Common Symptoms

- Corruption can cause 0F4 ABENDs
  - Corruption of the PDSE data set causes logical errors
  - Also may indicate an extent mismatch if the PDSE was moved
- Varied symptoms make improper sharing hard to diagnose
- Many symptoms can be caused by other issues

Improper PDSE Sharing: Admins Beware!

- There is no 100% safe way to circumvent EXTENDED mode's serialization requirements
- PDSE data sets cannot be serialized by third party products
  - Specifies RNL=NO
  - MIM does not serialize PDSEs
- Asking users not to update PDSEs from outside the SYSPLEX
  - Inevitably someone forgets
  - New users may not know the rules
- Reserves can cause serialization deadlocks

NORMAL Sharing Mode: Basics

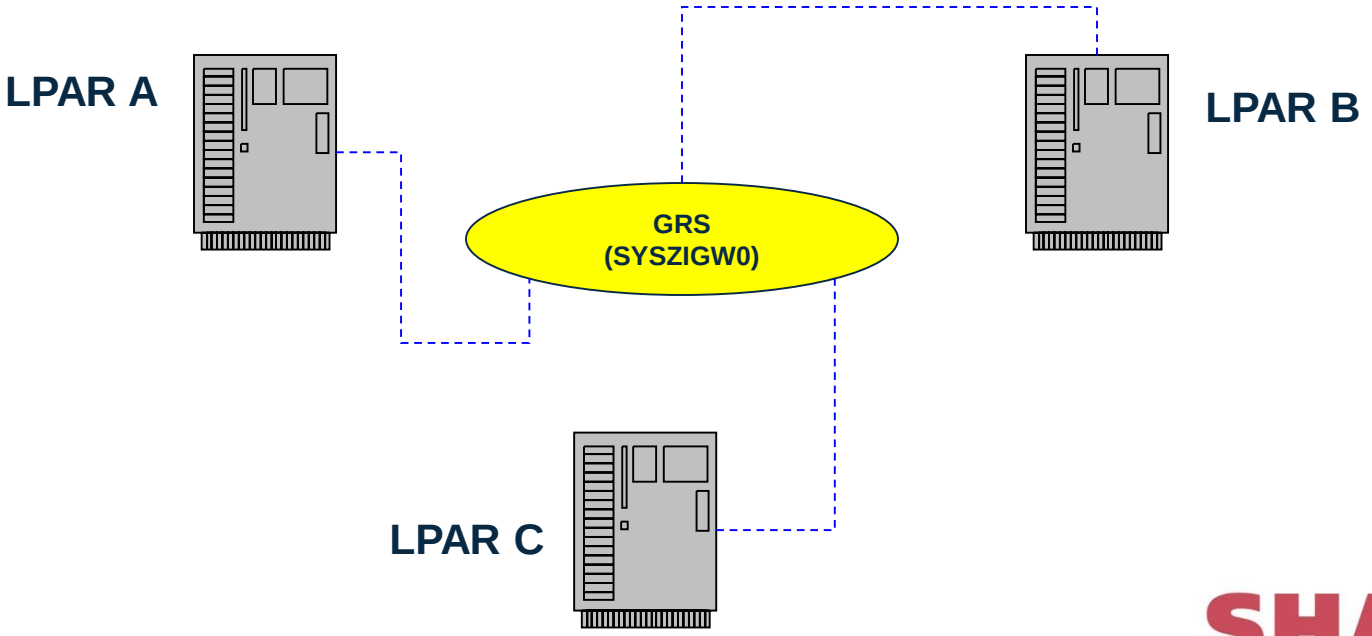
- Legacy PDSE sharing mode
- Provides the ability to share at the data set level between systems
- Shares at the member level on a single system
- Can only be implemented with the non-restartable address space (SMSPDSE)

NORMAL Sharing Mode: Startup

- PDSESHARING(NORMAL) specified in IGDSMSxx member
- NORMAL is the default sharing mode
- Mixed sharing modes are not supported
- To change from EXTENDED sharing to NORMAL sharing requires a SYSPLEX IPL

NORMAL Sharing Mode: Sharing Requirements

- NORMAL sharing is not limited to systems within the same SYSPLEX
- Participating systems must belong to the same GRSPLEX



NORMAL Sharing Mode: Sharing outside the SYSPLEX correctly

- Why it works:
  - NORMAL mode sharing only utilizes GRS for serialization
  - Multiple SYSPLEXs or stand alone LPARs may share DASD within the same GRSPLEX

- Limitations:
  - Restricts inter-system sharing to the data set level
  - When a system opens the PDSE for OUTPUT it is the only system that can access the PDSE
  - Can decrease performance by blocking opens of the data set

Sharing Mode

- **Does our process and configuration allow us to use Extended mode sharing?**

Decision: Recoverability

The Big Question

- Can we run the Restartable address space?
  - Requires Extended mode sharing

- The Restartable address space is a no-brainer
  - If you can run it, run it.
  - Benefits outweigh almost any effort needed to accommodate the move to Extended sharing

PDSE Recoverability: SMSPDSE1

- SMSPDSE1 is the restartable PDSE address space
  - SMSPDSE1 handles local data set connections
  - SMSPDSE handles global data set connections

- Enabled by IGDSMSxx Parameter
  - `PDSE_RESTARTABLE_AS(NO | YES)`

PDSE Recoverability: SMSPDSE1

- Why perform a SMSPDSE1 restart?
  - To recover from a situation that would otherwise require an IPL
  - Recover from a PDSE latch hang situation
  - Recover from in-core corruption of a PDSE at 1.12 and below
    - 1.13 and above can use the REFRESH command
  - Recover from excessive PDSE storage usage
- What are the side effects?
  - A small amount of CSA is lost in the restart

PDSE Recoverability: The SMSPDSE1 Restart Process

- Restart Warnings:
  - Do not route the restart command around the SYSPLEX
    - Each LPAR must complete it's restart before restarting the next
  - Depending on the number of connections that need to be quiesced and reconnected it may take a few minutes
  - Some user jobs may not be able to correctly handle the quiesce and reconnect processing and may fail

Decision: Buffer Beyond Close

Buffer Beyond Close

- What does Buffer Beyond Close do?
  - When DISABLED, the last close discards all buffered pages
  - When ENABLED, buffered pages are retained until discarded by LRU processing
  - Applies to Directory pages (and member pages if HIPERSPACE Caching is enabled)

- Performance Implications
  - Can potentially significantly reduce I/O
  - Enhances performance in cases where PDSE is frequently opened and closed.
  - Can increase overall storage utilization in some situations

Buffer Beyond Close

- IGDSMSxx Parameter
  - PDSE_BUFFER_BEYOND_CLOSE(YES|NO)
  - PDSE1_BUFFER_BEYOND_CLOSE(YES|NO)

- Caution:
  - Running with BBC enabled and using DCOLLECT on large numbers of volumes can result in significant storage growth
  - APAR OA47035 Prevents pages cached due to DCOLLECT from being retained beyond close

Decision: PDSE Version 2

PDSE V2 and Member Gens

- At z/OS 2.1 PDSE introduced the Version 2 Format
  - Improves performance and index efficiency
  - Improves partial release
  - Reduces CPU and Storage utilization
  - Allows for member generations

- Works exactly like a V1 PDSE
  - Pre-2.1 releases with toleration maintenance can access V2 PDSE's but not create them
  - 2.1 and above can fully use V2
  - V2 is expected to become the default format **eventually**

PDSE V2

- The PDS to PDSE conversion for COBOL can be a good time to go to V2
  - Take an entire set of libraries to V2 at once
  - No piecemeal conversion
  - Allows for use of Member Generations

- Version 2 is a streamlining of the format
  - Not a wholesale change
  - Overall structure and access is unchanged
  - Removes unused structures/unneeded layers of abstraction

Decision: PDSE Hiperspace Caching

PDSE Hiperspace Caching

- Optional extension of PDSE buffer management
  - Without Hiperspace caching PDSE only buffers directory pages
  - Hiperspace caching buffers member pages

- Disabled by default
  - Controlled/enabled by HSP_SIZE parameter
  - Uses the PDSE BMF LRU

PDSE Hiperspace Caching

- To be cache eligible:
  - PDSEs must have an MSR (Millisecond Response Time) of <10
  - Or caching must be requested by LLA
- Cached data is valid only so long as the dataset is open
  - Unless Buffer Beyond Close is enabled

- Effective for datasets which are held open for long periods and read by multiple jobs

Converting from PDS to PDSE

Considerations When Converting: Load Modules and Program Objects

- Load modules can always be converted to Program Objects

- Program Objects may not always be able to be converted to Load Modules
  - Due to Program Object features that are unsupported by Load Modules

- Cannot compare Load Module size to Program Object Size

IEBCOPY

- Simply use IEBCOPY to copy the load modules to a newly allocated PDSE
  - Conversions are automatically done by IEBCOPY
  - The conversion process will take some additional processing
    - Compared to a simple member copy
    - Program Objects requiring a pass through the binder
  - COBOL V5's binder output is a Program Object so no conversion is needed

- The actual conversion is simple
  - Process has been around for many years
  - The difficulties will be keeping track of all the libraries that require conversion

Caution: PDS Hygiene

- If you have load modules and data members in a PDS this can cause issues
  - PDS does not care if you mix members
  - PDSE requires that all members either be data or program objects
  - PDSE's will be "typed" by the first member created
  - "Type" does not change even if all members are deleted

- APAR OA46499
  - Prevents the IEBCOPY from failing in the event binder encounters a data member
  - Continues to process all load modules in the PDS

Adjusting to PDSE Libraries

Linklist Considerations

- Replacing a PDS in linklist versus a PDSE
  - For PDS's the following will work:

```
SETPROG LINKLIST,UNALLOCATE
P LLA
RENAME YOUR.LINKLIST.DATASET to
YOUR.LINKLIST.DATASET.OLD
RENAME YOUR.LINKLIST.DATASET.NEW to
YOUR.LINKLIST.DATASET
SETPROG LINKLIST,ALLOCATE
S LLA,SUB=MSTR
```

- This can cause 0F4 ABENDs if attempted with a PDSE

Linklist Considerations

- The correct way to update/replace a PDSE in linklist:
  - Cannot be updated while in an active LNKLST set
- To remove a data set from an active LNKLST set:

```
LNKLST DEFINE NAME(NEWLLSET) COPYFROM(OLDLLSET)
LNKLST DELETE NAME(NEWLLSET) DSNAME(data set.to.be.removed)
LNKLST ACTIVATE NAME(NEWLLSET)
LNKLST UPDATE JOB(*)
```

- Then re-add the updated data set
- This needs to be done on all LPARs sharing the linklisted PDSE

PDSE's, COBOL v5, HIPERSPACE, and you!

- Normally we'd expect our LLA managed program objects to be eligible to be cached via VLF
- COBOL V5 Program Objects use deferred segments
  - This makes them ineligible for caching in VLF **prior to OA45127** even if otherwise cache-worthy
  - In these cases LLA tells PDSE to cache the Program Object
  - For PDSE to cache the Program Object HIPERSPACE caching must be enabled
  - Without HIPERSPACE caching enabled or OA45127 applied COBOL V5 Program Objects WILL NOT BE CACHED

PDSE's, COBOL v5, HIPERSPACE, and you!

- **Hiperspace Caching Overview:**
  - Technote

- Hiperspace users at 2.1 should pick up OA46328
  - Corrects a space utilization issue

- Hiperspace users should also pick up OA47209
  - Improves the HSPSTATS output

Pending Deletes

- What is a Pending Delete?
  - When a member that is in-use is deleted
  - Member is no longer accessible but data and control blocks remain
  - Flagged for deletion during Pending Delete cleanup processing

- How do I know if I have an issue with Pending Deletes?
  - IGW01177T OUT OF SPACE CONDITION ENCOUNTERED DURING MEMBER CREATE
  - Growth in PDSEs utilization percentage without adding additional members
  - Utilization percent takes into account both valid member data and pending deletes

Pending Deletes

- Solution: APAR OA47755 (2.1 and 2.2)
  - IEBPDSE will now give a count of pending deletes
  - IEBPDSE will now allow you to manually run pending delete cleanup using the PerformPendingDelete option
  - Your IGDSMSxx can now have a parameter to run pending delete cleanup at a set time interval (on the next open for output)

- The best thing since the REFRESH command

PDSE Maintinence

- Get Current and Stay Current!
  - Unlike PDS, PDSE is still under active development
  - Apply HIPERs in a timely manner
  - We strongly recommend getting as current as possible prior to starting a move to PDSEs

Questions? Comments?

PDSE Requirements Survey

- PDSE is looking for your feedback!

- Survey and item descriptions are in the handout section for this presentation.

Please Fill Out the Survey!

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

Appendix

- Cheat Sheets, Parameters, Commands and JCL

NORMAL and EXTENDED Sharing Mode: Cheat Sheet

- Normal Sharing is the Legacy PDSE Sharing mode
  - It provides the ability to share at the *dataset* level between systems.
    - It can share at a *member* level only within a single system.
  - It can only be implemented with the Non-Restartable Address Space (SMSPDSE).
    - If you wish to use the restartable Address Space you **must** use Extended Sharing.
  - Getting started with Normal Sharing
    - PDSESHARING(NORMAL) must be specified in the IGDSMSxx member.
    - In order to change from a Extended Sharing Mode to Normal Sharing Mode, you **must** IPL.
  - Normal Sharing Mode is not limited to systems in the same SYSPLEX.
    - However, all participating systems must belong to the same GRSPLEX

- Extended Sharing is the preferred method of sharing
  - It provides the ability to share at the *member* level between systems.
  - It works with either and/or both of the SMSPDSE Address Spaces active.
  - Getting started with Extended Sharing
    - PDSESHARING(EXTENDED) must be specified in the IGDSMSxx member.
    - The SYSPLEX sharing type is determined by the first PDSE Address Space to start.
    - IPL is recommended to start Extended Sharing.
      - ACTIVATE Command can be used, but may cause PDSE problems.
  - Extended Sharing is strictly limited to systems within the same SYSPLEX.
    - All participating systems must belong to the same GRSPLEX AND XCFPLEX

NORMAL and EXTENDED Sharing Mode: Cheat Sheet

- Normal Sharing is the Legacy PDSE Sharing mode
  - It provides the ability to share at the *dataset* level between systems.
    - It can share at a *member* level only within a single system.
  - It can only be implemented with the Non-Restartable Address Space (SMSPDSE).
    - If you wish to use the restartable Address Space you **must** use Extended Sharing.
  - Getting started with Normal Sharing
    - PDSESHARING(NORMAL) must be specified in the IGDSMSxx member.
    - In order to change from a Extended Sharing Mode to Normal Sharing Mode, you **must** IPL.
  - Normal Sharing Mode is not limited to systems in the same SYSPLEX.
    - However, all participating systems must belong to the same GRSPLEX

- Extended Sharing is the preferred method of sharing
  - It provides the ability to share at the *member* level between systems.
  - It works with either and/or both of the SMSPDSE Address Spaces active.
  - Getting started with Extended Sharing
    - PDSESHARING(EXTENDED) must be specified in the IGDSMSxx member.
    - The SYSPLEX sharing type is determined by the first PDSE Address Space to start.
    - IPL is recommended to start Extended Sharing.
      - ACTIVATE Command can be used, but may cause PDSE problems.
  - Extended Sharing is strictly limited to systems within the same SYSPLEX.
    - All participating systems must belong to the same GRSPLEX **AND** XCFPLEX

Common Pitfalls of Sharing: Cheat Sheet

- Sharing a PDSE outside of the XCFPLEX
  - By sharing a PDSE outside of the XCFPLEX in Extended Sharing, you are introducing unpredictable problems.
  - Corruption can cause 0F4 ABENDs
    - Varied symptoms make improper sharing harder to diagnose
  - Improper sharing can result in unserialized access to datasets.
    - There is **no** warning that a dataset has been accessed in this manner.
    - Potential issues:
      - Invalid index data in-core, Corrupt dataset on DASD, corrupt member data, mismatched extent data, or even nothing at all.
  - There is no sure fire way to circumvent Extended Sharing Mode's serialization requirements.
    - PDSE datasets cannot be serialized by 3rd party products.
    - Asking users not to update PDSEs from outside SYSPLEX
      - Too hard to enforce, inevitably someone forgets, new users may no know all the rules
    - Reserves can cause serialization deadlocks.

Appendix:
Parameters, Commands and JCL

- PDSE Console Dump Parameters

```
COMM=(PDSE PROBLEM)
 JOBNAME=(*MASTER*,SMSPDSE*),
 SDATA=(PSA,CSA,SQA,GRSQ,LPA,LSQA,RGN,SUM,SWA,TRT,COUPLE
 ,XESDATA),END
```

- IGDSMSxx Parameters:
  - SMSPDSE1 restartable address space:

```
PDSE_RESTARTABLE_AS(NO | YES)
```

  - PDSE Sharing Modes:

```
PDSESHARING(EXTENDED|NORMAL)
```

  - PDSE Member Generations Installation Limit

```
MAXGENS_LIMIT=n
```

Appendix:
Parameters, Commands and JCL

- PDSE Console Commands
  - SMSPDSE1 Restart Command

```
V SMS,PDSE1,RESTART
  [,QUIESCE(duration | 15 )[,COMMONPOOLS(NEW|REUSE) ]
```

  - SMSPDSE1 Activate Command

```
V SMS,PDSE1,ACTIVATE
```

  - PDSE Analysis Command

```
V SMS,PDSE(1),ANALYSIS
```

  - PDSE Freelatch Command

```
V SMS,PDSE|PDSE1,FREELATCH(<latch address>,asid,tcb)]
```

Appendix:
Parameters, Commands and JCL

- IEBPDSE JCL (1.13 and above only)

```
//VALIDATE EXEC PGM=IEBPDSE
//SYSPRINT DD SYSOUT=*
//SYSIN DD DUMMY
//SYSLIB DD DISP=SHR,DSN=INPUT.PDSE.BAD
```

Appendix:
Parameters, Commands and JCL

- DSS PHYSICAL dump JCL

```
//DUMP      EXEC PGM=ADRDSSU
//SYSPRINT DD  SYSOUT=*
//OUT       DD  UNIT=3390,
//              VOL=SER=XXXXXX,
//              DISP=(NEW,KEEP),
//              SPACE=(CYL,(100,100)),
//              DSN=hilev.DSSDUMP,
//              DCB=BLKSIZE=32760                                  /
/SYSIN    DD  *
DUMP  PIDY(vvvvvv) -
     OUTDD(OUT)  -
      DATASET(INCLUDE(pdse.dataset.name)) -
      ALLDATA(  *  )                                              /
*
```

PDSE Recoverability: How to Restart SMSPDSE1

- Step 1:
  - Gather doc! At a minimum a console dump of SMSPDSE and SMSPDSE1 should be taken
- Step 2:
  - Issue the restart command
  - `V SMS,PDSE1,RESTART [,QUIESCE(duration | 15 )[,COMMONPOOLS(NEW|REUSE) ]`
  - QUIESCE option determines how long in-flight operations have to quiesce
  - COMMONPOOLS option determines whether ECSA cell pools are reused
    - Only select NEW if there was a cell pool problem

PDSE Recoverability: Phases of the SMSPDSE1 Restart

- Quiesce Phase
  - New PDSE requests are corralled
  - By default all in-flight activity has 15 seconds to complete
  - If requests do not complete within the quiesce interval the user has the choice to either wait or continue with the restart
  - Once the quiesce interval completes SMSPDSE1 stops and a new instance is started

PDSE Recoverability: Phases of the SMSPDSE1 Restart

- Reconnect Phase
  - All user connections are restored
  - There is a 15 second time limit on reconnect processing
  - If reconnect cannot be completed within 15 seconds the user can choose to retry for another 15 seconds or continue
  - Users must decide whether they can afford to lose any tasks which don't reconnect in a timely manner

Appendix:
SMSPDSE1 Restart Message Sequence

V SMS,PDSE1,RESTART

IGW036I VARY SMS,PDSE1,RESTART COMMAND ACCEPTED.
IGW057I WAITING FOR SMSPDSE1 SHUTDOWN.
IGW055I SMSPDSE1 SHUTDOWN IN PROGRESS.
IGW999I XQUIESCE Started
IGW062I SMSPDSE1 IS QUIESCING.

IGW064I SMSPDSE1 IGNORING IN-PROGRESS TASK 001B:MHLRES2B, TCB@=007DEC4 8.
***169 IGW074D SMSPDSE1 QUIESCE FAILED. RETRY? (Y/N)**

R 169,N
IEE600I REPLY TO 169 IS;N
IGW065I SMSPDSE1 QUIESCE COMPLETE.
IGW058I SMSPDSE1 SHUTDOWN COMPLETE.
IGW059I SMSPDSE1 IS BEING ACTIVATED.
IGW040I PDSE IGWLGEDC Connected
IGW040I PDSE Connecting to XCF for Signaling
IGW040I PDSE Connected to XCF for Signaling
IGW040I PDSE Posting initialization
IGW043I PDSE MONITOR IS ACTIVE 040
++ INVOCATION INTERVAL:60 SECONDS
++ SAMPLE DURATION:15 SECONDS

IGW061I SMSPDSE1 INITIALIZATION COMPLETE.

IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW066I SMSPDSE1 IS RECONNECTING ALL USERS.
IGW069I SMSPDSE1 RECONNECT PHASE COMPLETE.
IGW070I SMSPDSE1 WILL RESUME ALL USER TASKS.
IGW999I XQUIESCE Stopping

IGW999I Reconnect Completed Normally