Using PDSEs in your SYSPLEX: Best Practices and Troubleshooting

Speaker: Thomas Reed /IBM Corporation

SHARE Pittsburgh 2014

Session:16126

#SHAREorg

Agenda

- PDSE Sharing
- PDSE Recoverability
- PDS to PDSE Conversion Considerations
- Customer PDSE Issue Troubleshooting
  - Latch Hang/Contention
  - Corrupt PDSE Data Set

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 Sharing Basics

Important Terminology:

- Two sharing modes, NORMAL and EXTENDED
  - NORMAL is the default and fallback 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

EXTENDED Sharing Mode: Basics

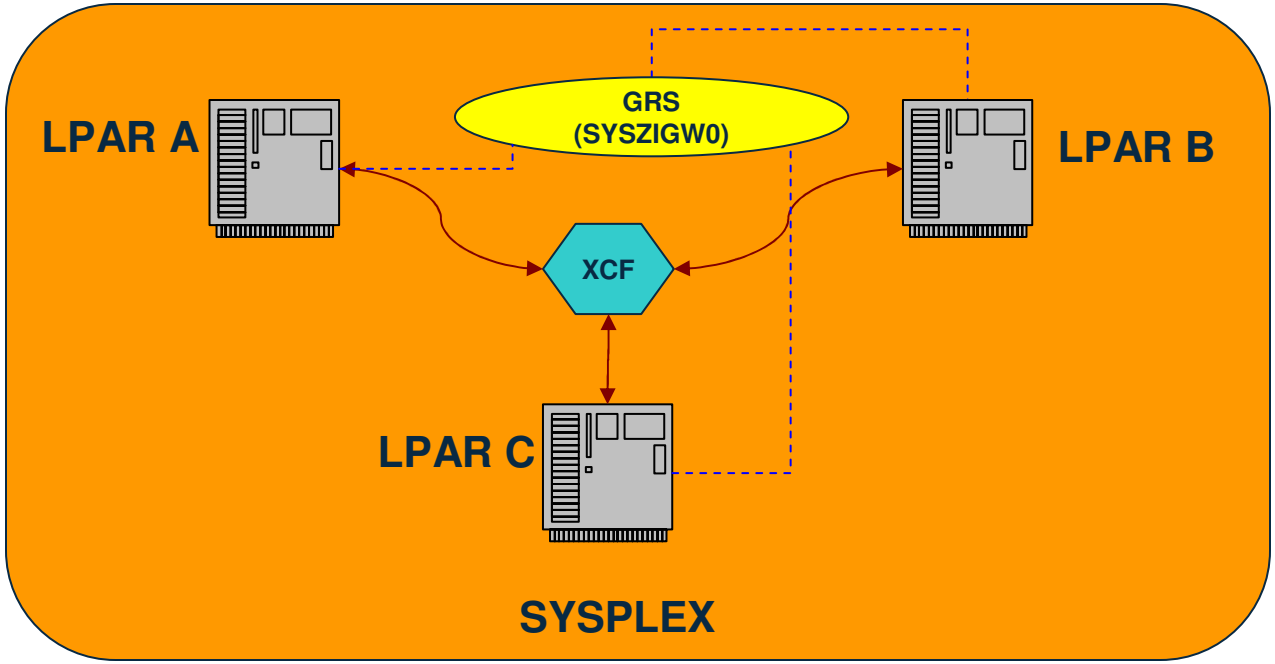
- The newest and preferred sharing mode
- Provides the ability to share at the member level between systems
- Can be implemented with one or both address spaces active

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

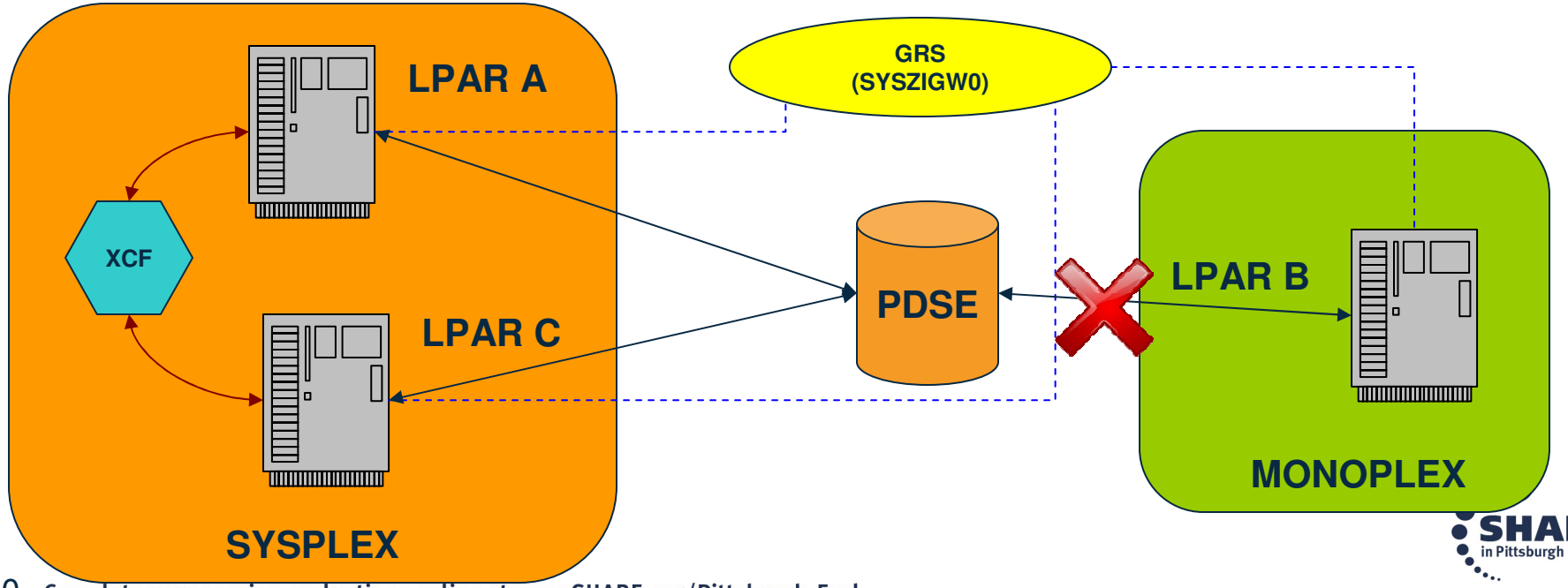


EXTENDED Sharing Mode: Recap

- EXTENDED sharing is the preferred sharing mode for PDSE
- PDSESHARING(EXTENDED) must be specified in IGDSMSxx member
- EXTENDED sharing is strictly limited to systems within the same SYSPLEX

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

Improper PDSE Sharing: Recap

- Sharing PDSEs outside of a single SYSPLEX while running EXTENDED sharing is unsupported
- Unserialized access to a PDSE causes a range of unpredictable effects
- There is no effective way to get around serialization restrictions while running EXTENDED sharing

NORMAL Sharing Mode: Basics

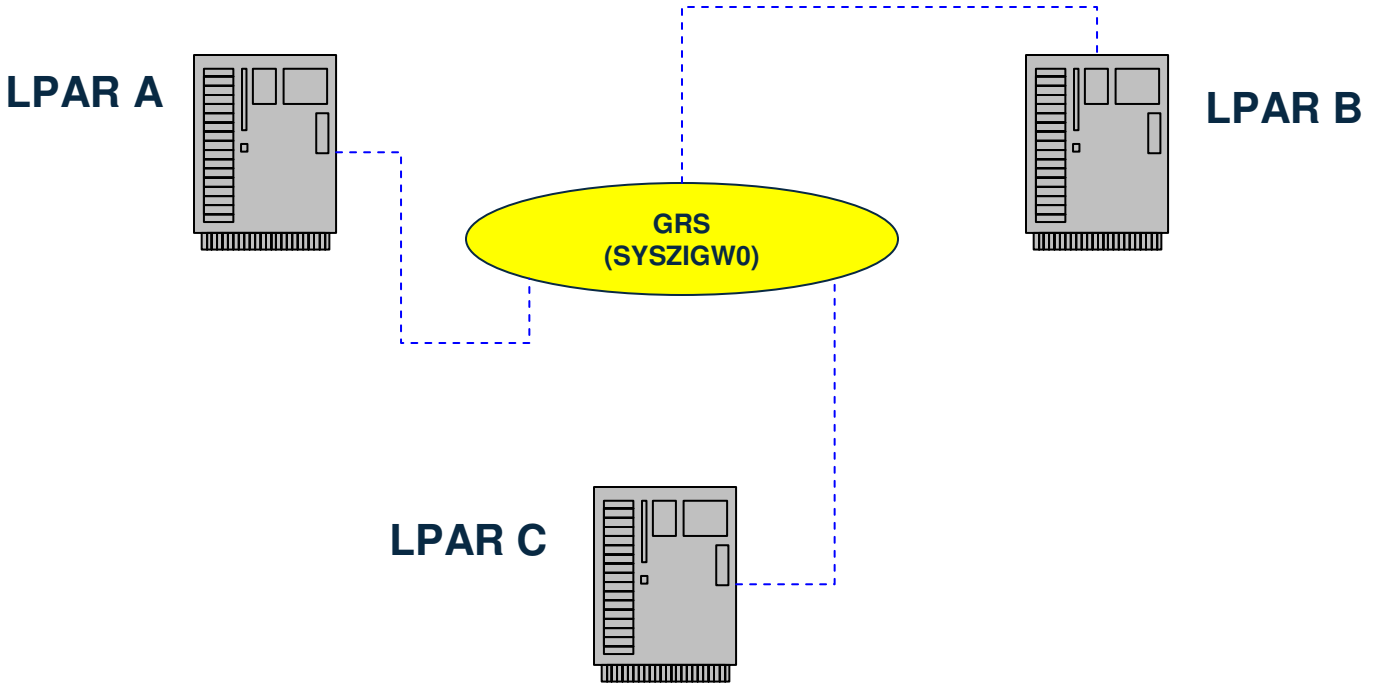
- Legacy PDSE sharing mode
- Provides the ability to share at the data set level between systems
- Shares at the member level on a single system
- Can only be implemented with the non-restartable address space (SMSPDSE)

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 an 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

NORMAL Sharing Mode: Recap

- NORMAL sharing is the historic (and default) sharing mode for PDSE
- `PDSESHARING(NORMAL)` must be specified in IGDSMSxx member
- NORMAL sharing is strictly limited to systems within the same GRSPLEX
- Allows PDSEs to be shared outside a single SYSPLEX at the data set level
- Data set level sharing can cause performance restrictions

PDSE Recoverability: SMSPDSE1

- SMSPDSE1 is the restartable PDSE address space
  - SMSPDSE1 handles local data set connections
  - SMSPDSE handles global data set connections
- SMSPDSE1 is only available to systems running PDSE sharing EXTENDED
- It is highly recommended that all customers running EXTENDED sharing take advantage of the restartable address space
- Enabled by IGDSMSxx Parameter
  - `PDSE_RESTARTABLE_AS(NO | YES)`

PDSE Recoverability: The SMSPDSE1 Restart Process

- 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

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

PDSE Recoverability: SMSPDSE1 Restart Failure

- If necessary tasks in SMSPDSE1 have been lost the restart may hang
  - Can result from ABEND S0C1 in the wrong task
- SMSPDSE1 then must be forced down
- The user can then attempt to use the ACTIVATE command to bring SMSPDSE1 back up
- `V SMS,PDSE1,ACTIVATE`
- This is a last resort and should only be used to attempt to avert an IPL

PDSE Recoverability: z/OS 1.13 New PDSE Commands

- The **REFRESH** command provides a more granular way to recover from in-core corruption at 1.13 and above
- Avoids a PDSE1 restart for corruption issues
- Discards all in-core pages for the specified PDSE
- Next access will re-read all data from DASD

- `V SMS,PDSE|PDSE1,REFRESH,DSN(dsname)[,VOL(volser)]`

- **WILL NOT** resolve PDSE latch/lock issues

PDSE Recoverability: z/OS 1.13 New PDSE Commands

- The **CONNECTIONS** command allows the user to discover where the PDSE is in use at 1.13 and above
- Useful for PDSE in use errors
- A dump no longer has to be taken to determine what tasks are connected to a PDSE

- `D SMS,PDSE|PDSE1,CONNECTIONS,DSN(dsname),<VOLSER(volser)>`

PDSE Recoverability: Recap

- When running EXTENDED sharing it is highly recommended that SMSPDSE1 be enabled
- SMSPDSE1 restart can help avoid the need to IPL due to PDSE problems
- 2 phases of the restart
  - Quiesce (default 15 second delay)
  - Reconnect (default 15 second delay)
- 1.13 and above commands
  - REFRESH command allows the user to purge in-core directory pages to resolve in-core corruption
  - CONNECTIONS command allows the user to determine the tasks connected to a PDSE without having to take a dump

Considerations When Converting: PDS to PDSE

- **PDS:**
  - Alphabetically organized linear directory consisting of member names and pointers to those members
  - Sequential areas of the data set are used for member data and not reused
- **PDSE:**
  - Collection of 4K pages. Both directory pages and member data
  - Multiple tree style directories pointing to linear data spaces for member data

Considerations When Converting: 2.1

- **The PDSE IMF/BMF code has been rewritten**
  - This results in significant performance improvements
  - Reduced CPU and Storage usage
  - Applys to both V1 and V2 PDSE's
- V2 PDSEs are available for creation
  - Specifying LIBRARY,2 will create a V2 data set
  - A streamlining of the V1 format
  - Offers additional performance improvements over V1
  - Enables PDSE Member Generations support

Considerations When Converting: 2.1

- At 2.1, the PDSE member record limit has been expanded
  - It is now larger than the largest allowable PDS
  - This means that anything you could fit in a PDS can now be fit into a single PDSE member
  - 2.1 now allows for ~2bn records per member
  - 1.13 and below only allows ~15m records per member
- IEBPDSE is enhanced
  - Recommend running IEBPDSE from the the highest level system available
  - Due to the IMF rewrite, do not expect IEBPDSE at 2.1 to return identical results to 1.13

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
- Conversions are automatically done by IEBCOPY
- The conversion process will take additional processing due to Program Objects requiring a pass through the binder
- Cannot compare Load Module size to Program Object Size

Considerations When Converting: 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 will cause 0F4 ABENDs if attempted with a PDSE

Considerations When Converting: Linklist Considerations

- The correct way to replace a PDSE in linklist:

To remove a PDSE in LNKLST:

```
SETPROG LNKLST,DEFINE,NAME=LNKLST2,COPYFROM=LNKLST1
SETPROG LNKLST,DELETE,NAME=LNKLST2,DSNAME=YOUR.LINKLIST.PDSE
SETPROG LNKLST,ACTIVATE,NAME=LNKLST2
SETPROG LNKLST,UPDATE,JOB=*
F LLA,REFRESH
```

To add it back, do the following:

```
SETPROG LNKLST,ADD,NAME=LNKLST2,DSNAME=YOUR.LINKLIST.PDSE
SETPROG LNKLST,ACTIVATE,NAME=LNKLST2
SETPROG LNKLST,UPDATE,JOB=*
F LLA,REFRESH
```

- This needs to be done on all LPARs sharing the linklisted PDSE

Considerations When Converting: Recap

- PDSs and PDSEs are similar in interface but are entirely different internally
- At 2.1 PDSE members can now contain anything a PDS can
- Program Objects offer additional features over Load Modules
- Conversion from Load Module to Program Object takes additional processing
- Attempting to replace a PDSE in linklist like a PDS will result in problems

Customer PDSE Troubleshooting

- Latch Hangs/Latch Contention
- Corrupt PDSE Data set

Customer PDSE Troubleshooting: Latch Hangs/Latch Contention

- **Symptoms:**
  - Hung job(s), non-responsive TSO sessions
  - IGW038A messages
    - `IGW038A POSSIBLE PDSE PROBLEM(S). (SMSPDSE|SMSPDSE1)`
    - Issue the analysis command: `V SMS,PDSE(1),ANALYSIS`
    - It is recommended that the analysis command be issued by automation
    - Be sure to issue the analysis against the correct address space
    - Analysis command results in an IGW031I message

Customer PDSE Troubleshooting: Latch Hangs/Latch Contention

```
IGW031I PDSE ANALYSIS  Start of Report(SMSPDSE1)
        ---------data set name---------------------- -----vsgt-------
        HLQ.TOM123.JCL                               01-THEVOL-12345E
        ++ Unable to latch HL1bPlch:000000007E7A3EE0
          Latch:000000007E7A3F10 Holder(019C:009F8BE8)
          Holding Started Task:MYJOB
        ++ Lock GLOBAL DIRECTORY SHARED
          held for at least 19 seconds
          Hl1b:7E7A3EE0 HOLDER(019C:009F8BE8)
          Holding Started Task:MYJOB
        PDSE ANALYSIS  End of Report(SMSPDSE1)
```

Holder ASID:TCB

Held Resource

Customer PDSE Troubleshooting: Latch Hangs/Latch Contention

- Contention vs. Hang
  - Contention: Resource holders will change as may the held resources
    - May be caused by slow job completion due to IO, low CPU resources, waiting on other resources etc.
    - Will eventually clear on it's own
  - Hang: Resource holder will not change although waiters may
    - Result from hung tasks, orphaned resources, looping jobs, suspended tasks etc.
    - Require user action to resolve

Customer PDSE Troubleshooting: Latch Hangs/Latch Contention

- First, gather doc:
  - SVC Dump of SMSPDSE (and SMSPDSE1)
  - LOGREC and SYSLOG starting before the first IGW038A message
- Second, determine the state of the holder:
  - If the task is still in the system and suspended/hung then cancel it if possible, force if necessary
  - If the task is out of the system but is still shown as the holder then the latch is orphaned
- Finally, free the latch:

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

Customer PDSE Troubleshooting: FREELATCH Notes

- ASID and TCB of zeros are valid
- FREELATCH will check the latch address to make sure it is actually a held latch
- It is important that the latch holder be out of the system before using FREELATCH
  - If the holder is still active and a FREELATCH is issued it could lead to data set corruption

Customer PDSE Troubleshooting: Locking Notes

- Because PDSE shares between systems, serialization issues may also occur between systems
- In IGW031I messages you may see:
- `++ Message to SYS1    .SMSPDSE1 pending for 652 seconds`
- This indicates a delay in messaging between PDSE address spaces between systems
  - Usually occur in conjunction with long held locks
  - Can indicate linked PDSE issues on other systems
- It is recommended that dumps of the PDSE address spaces be taken on the system that is not responding to messaging, e.g. SYS1

Customer PDSE Troubleshooting: Corrupted PDSE Data Set

- PDSE Corruption
  - PDSEs contain both Directory data and Member data
  - We can only detect corruption of the PDSE Directory
  - 2 types of PDSE Directory corruption, in-core and hardened to DASD
- Symptom:
  - ABEND 0F4 on access of the PDSE
  - Reason codes will vary depending on what specifically went wrong
  - There may be multiple ABEND 0F4's associated with one failed access. The first one is the most important one for L2 diagnosis

Customer PDSE Troubleshooting: Corrupted PDSE Data Set

- In-core corruption:
  - Only the buffered pages/control structures associated with the PDSE are corrupt
  - Because each PDSE address space maintains it's own buffers the corruption is localized to a single system
  - Can be checked for by accessing the PDSE from another system in the SYSPLEX
  - Correctable via an SMSPDSE1 restart (if the corruption is in the SMSPDSE1 address space)
  - At 1.13 and above the REFRESH command can be used in lieu of a SMSPDSE1 restart (see slide 26)

Customer PDSE Troubleshooting: Corrupted PDSE Data Set

- Corruption hardened to DASD
  - The PDSE on DASD is corrupt
  - At 1.13 and above the IEBPDSE utility can validate the state of the PDSE on DASD (see Appendix for JCL)
    - Due to the IMF/BMF rewrite at 2.1, IEBPDSE may produce different reason codes at 2.1 than 1.13
  - All systems in the PLEX will encounter the same errors when accessing the data set
  - Data set will need to be recovered from the last backup
  - A DSS PHYSICAL dump of the data set should be collected for L2 (see Appendix)

Customer PDSE Troubleshooting: Corrupted PDSE Data Set

- Documentation to gather:
  - SVC Dump(s)
  - LOGREC and SYSLOG going back to the last good access of the data set
  - Retain the corrupt copy of the data set
  - Take a DSS PHYSICAL dump of the corrupt data set (see Appendix)

Questions? Comments?

Please Fill Out the Survey!

Appendix

Parameters, Commands and JCL

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(  *  )
/*
```

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