Monitoring a data sharing system can be a daunting, time-consuming task. But there is a real need to keep track of how your data sharing system is running. This presentation discusses the major mechanisms of group buffer pools and global locking, and includes the major points of monitoring these mechanisms for performance. User tips and tricks are included to assist you in monitoring your data sharing system.

Presentation Outline

- **Global Buffering Setup and Monitoring**
- **Monitoring Group Buffer Pools**
- **Global Locking Mechanics**
- **Monitoring Global Locking**
- **User Tips on Monitoring Data Sharing Groups**

2

These are the major points to be covered in this presentation. In the first hour, some introductory remarks will be made about getting started in data sharing, focused primarily on those customers who are just getting started. However, this section will also point out definitions that even experienced data sharing users should double-check.

The remainder of the presentation will cover the mechanics of global buffering and global locking used by DB2 for z/OS data sharing, and the monitoring points for checking the performance of those mechanisms.

As appropriate, user tips and tricks will be included to assist the user in the task of monitoring data sharing performance.

Getting Started with Data Sharing

- **DB2 CF Structures Need to be Defined**
- **Sizes Need to be Estimated**
- **Suggestions in DB2 Manuals**
- **CF Sizer Available**
  - **Assumes Latest Hardware**
- **Policy Needs to be Intialized**

3

One of the first steps for an installation moving to a data sharing environment, after planning a naming convention, is to estimate and plan for the size and placement of the DB2 structures.

Suggestions are included in the data sharing administration manual, and an online size tool is available. **NOTE**: the CF Sizer tool assumes the latest version of software and hardware.

When the initial names, sizes, and placements have been decided, a CFRM (Coupling Facility Resource Manager) policy must be initialized and put in place in the z/OS system.

Getting Started with Data Sharing . . .

- **IXCL1DSU** Utility Creates the CFRM Couple Data Set
  - Maximum Number of Policies
  - Maximum Number of Connections
  - Maximum Number of Structures
- **IXCMIAPU** Utility Creates the Policy
- These z/OS Utilities are Usually Responsibility of z/OS Support Staff

4

A z/OS utility called IXCL1DSU is run to format the CFRM couple data set, which is used to store CFRM policies. Policies are CF structure definitions used to set up and run a parallel sysplex, and are stored on the CFRM couple data set. This set of definitions is processed by a z/OS utility called IXCMIAPU.

These two utilities are usually the responsibility of the z/OS support staff, with strict security protecting them (the utilities, not the staff.)

An installation needs to work out the operational steps necessary to allow the DB2 staff to influence the settings of the policy for DB2 structures.

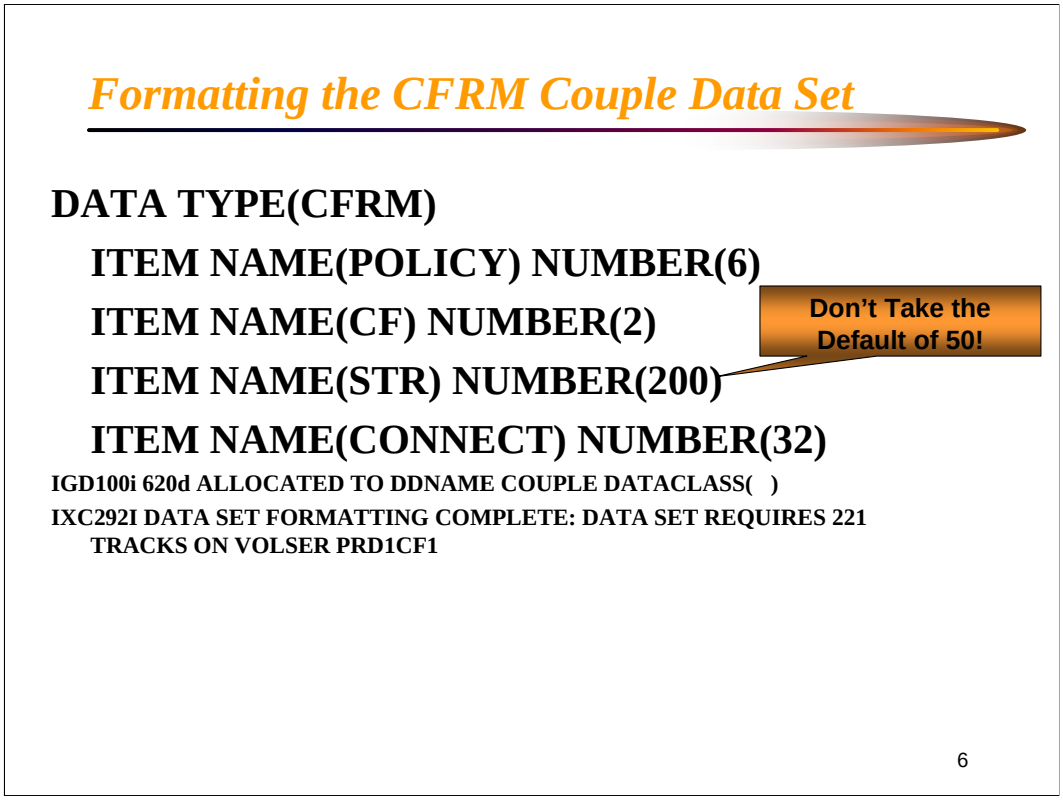
Defining CF Structures - the Policy

- Set of Definitions for z/OS
- Created by **IXCMIAPU** Utility
- Stored on a Couple Data Set
  - Formatted by **IXCL1DSU** Utility
    - Maximum Number of Policies
    - Maximum Number of Connections
    - Maximum Number of Structures
- Activated with a "**SETXCF START** …." Command

5

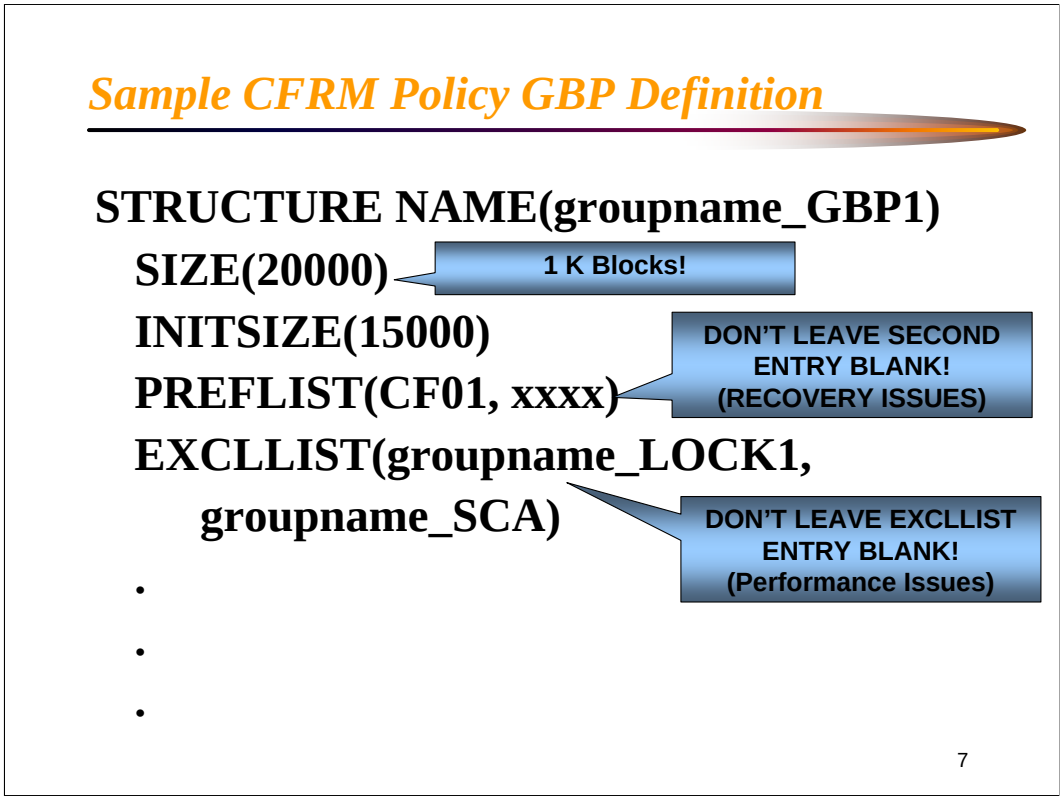
Definitions used to set up and run a parallel sysplex are stored in a set of system parameters called a policy. This set of definitions is processed by an z/OS utility called IXCMIAPU. Its output is stored on the couple data set, which was formatted with another z/OS utility called IXCL1DSU.

A newly created CFRM policy must be activated by an operator command before new definitions for structures will take effect. Further, as we will see in a subsequent visual, any definition changes to an existing allocated structure in the CF will not take effect until the structure is rebuilt by the user.



The z/OS utility IXCL1DSU formats the CFRM couple data set, which is where the CFRM policies with structure definitions are stored.

When first formatting the CFRM CDS (Couple Data Set), make sure the defaults are not taken, especially for the number of structures. The default value is 50 – and in any size parallel sysplex, 50 is a very small number. Plan for where the parallel sysplex may be in a couple of years, not what the initial configuration is!



All DB2 structures begin with "groupname", which is defined in the planning stage of data sharing, and chosen during installation of data sharing.

SIZE - This is the maximum size, in 1K blocks, to which this structure can increase with this policy active. **NOTE**: Don't forget to estimate size correctly – 1K blocks here vs. 4K blocks for local buffer pools!

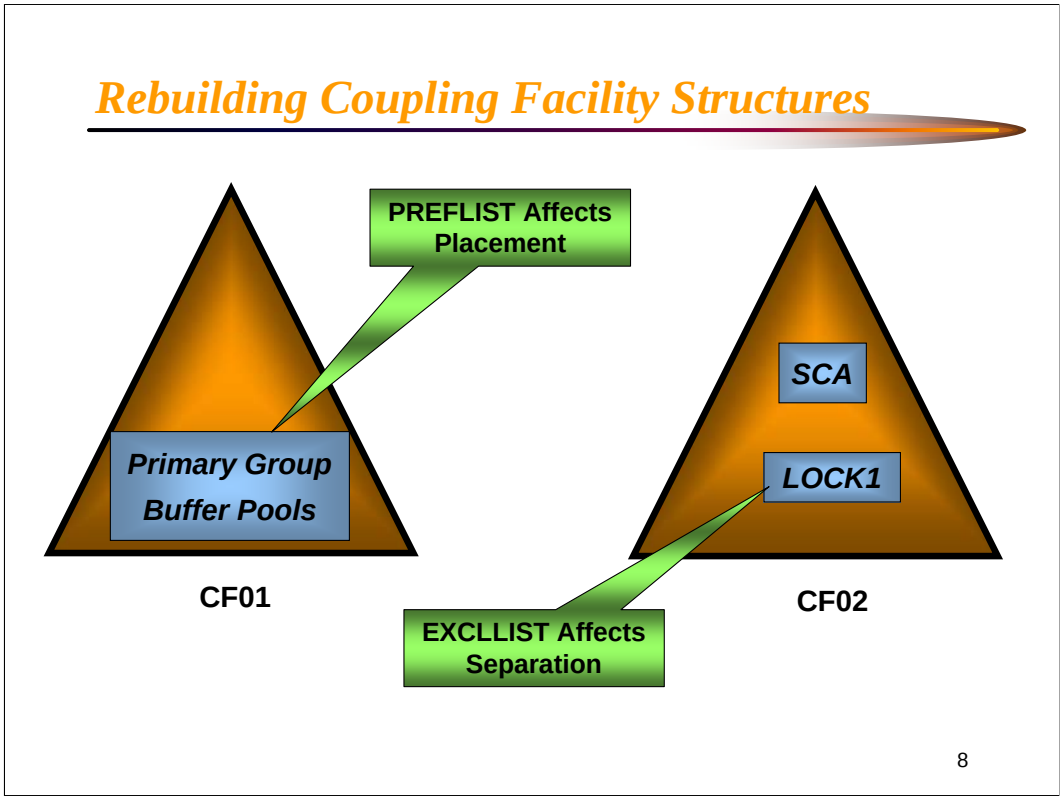
INITSIZE - This is the size at which the structure will be allocated by DB2 with the first CONNECT, or by a manual REBUILD command.

PREFLIST - This is a list of the preferred CFs for z/OS to allocate this structure.

NOTE: This is only a preference list. XES also looks at other factors, such as failure independence, connectivity, volatility, etc., before considering the user preferences.

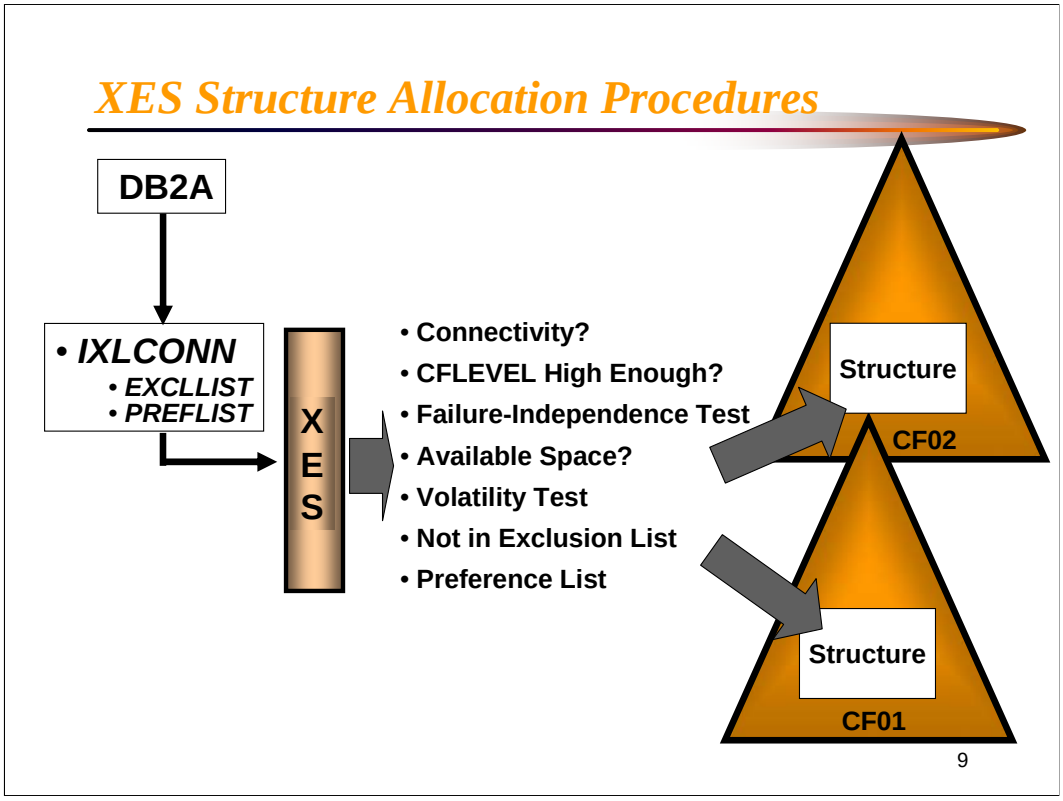
NOTE: The second specification is also important for recovery – if the CF in the first parameter fails, the system will look to the CF specified in the second parameter to determine where to rebuild the structures. If blank, no rebuilds are done!

EXCLLIST - This lists the structures that should NOT be in the same CF as the one being defined. In general, the LOCK1 and SCA structures should be in a different CF than the primary group buffer pool.



The CFRM policy parameters PREFLIST and EXCLLIST influence the allocation of the DB2 structures by XES.

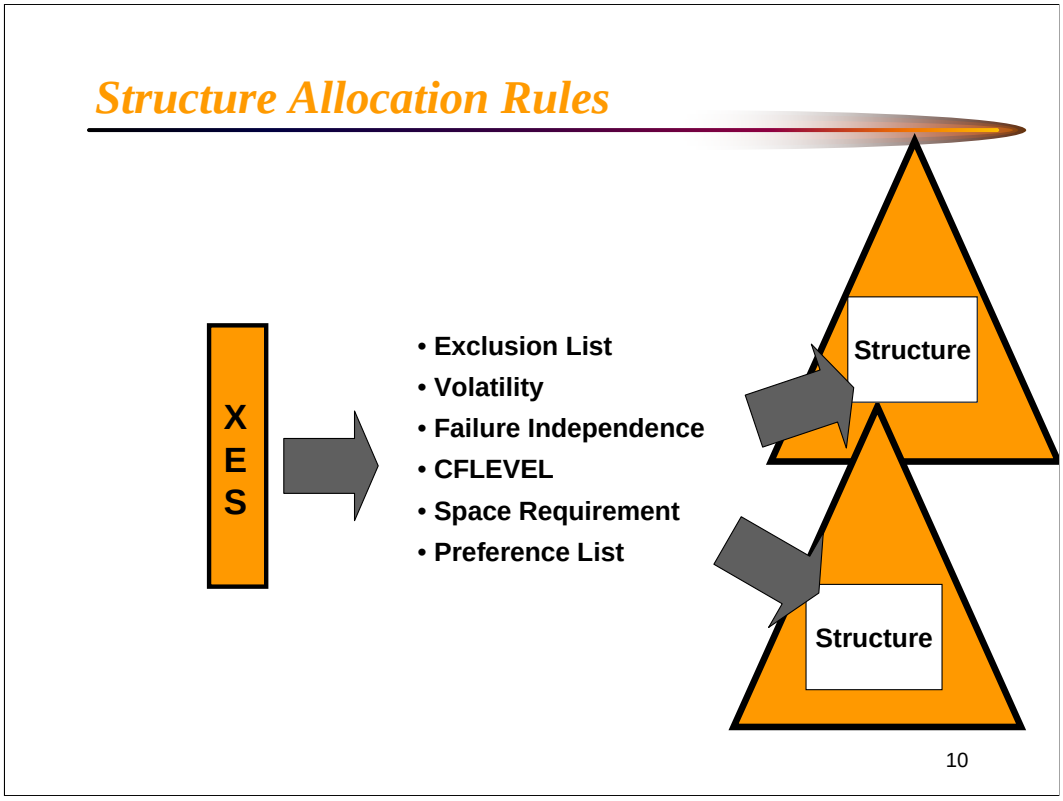
PREFLIST indicates the user's desired location of the structure, while EXCLLIST indicates which structures are not to be allocated along with the structure being defined.



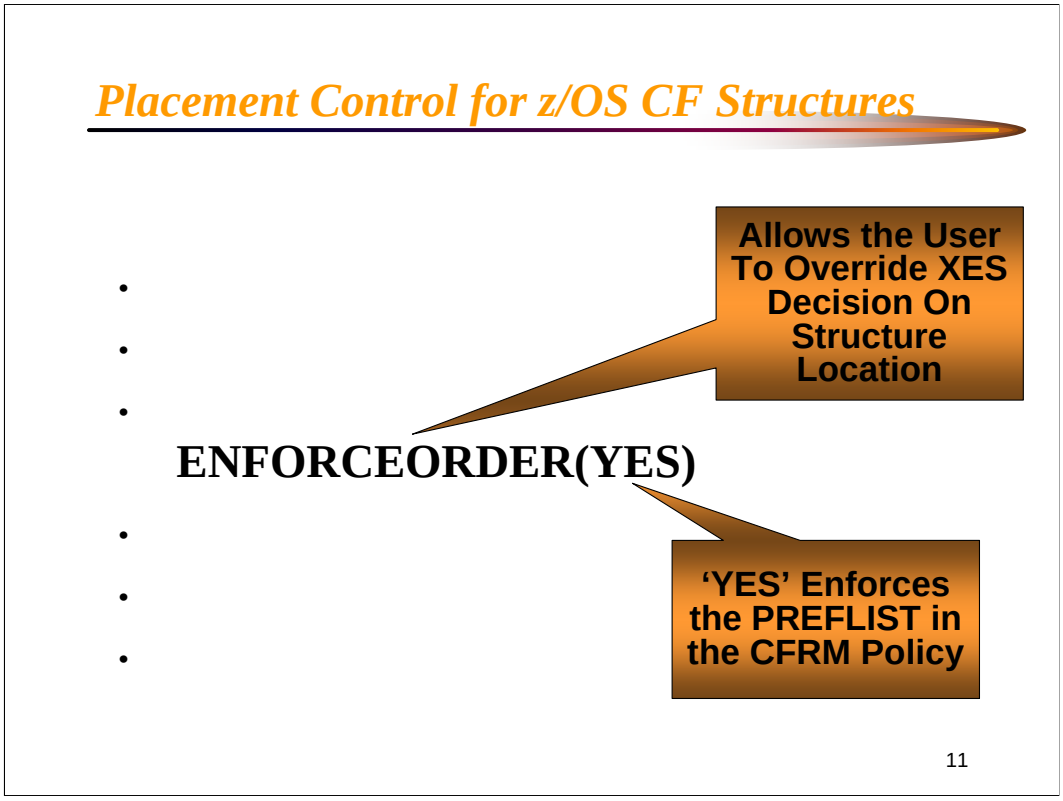
When a DB2 or IRLM first requires access to a structure, it will issue an internal macro, **IXLCONN**, to allocate or connect to the structure. Among other parameters, this macro will pass the contents of the CFRM policy definitions for the structure, such as the name, size, and preference and exclusion lists.

However, XES looks at other factors in determining placement of the structures. It checks, in the order shown on the visual, for physical connectivity to the requested CF, as well as required CFLEVEL of CFCC code. It looks at the failure dependence attribute of the CFs on the system. It, of course, looks at available space. The volatility of the CF (presence or absence of UPS (Uninterruptible Power Supply)) is also considered. All of these checks are performed **BEFORE** the user's wishes are considered, as specified in the **PREFLIST** and **EXCLLIST**. These internal checks are important to understand, as XES, depending on the hardware setup of the CFs on a system, may sometimes put structures in the **WRONG CF**! (At least wrong according to the PREFLIST!).

If XES does allocate structures in the 'wrong' CF, check these other factors to determine possible reasons for the 'misallocation'.

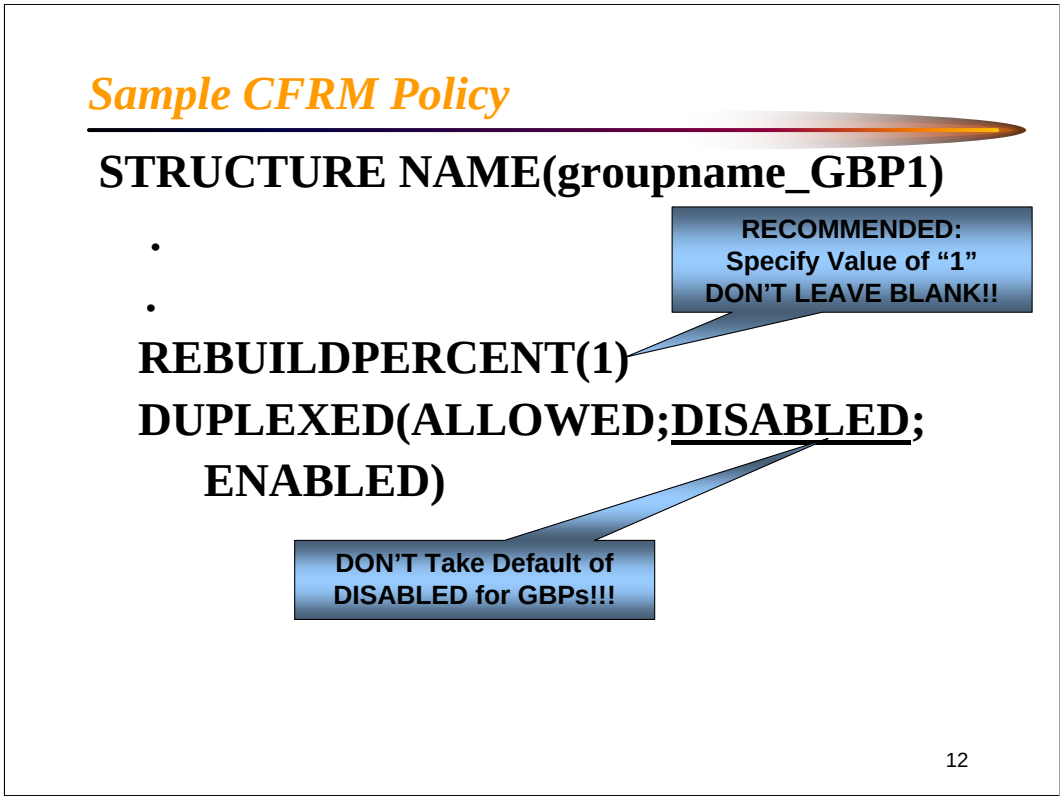


If XES is unable to allocate the structure after the checks on the previous visual, it will work through the list backwards, until it can allocate the structure. Depending on what caused XES to override the parameters in the PREFLIST and EXCLLIST, the resulting allocation may not be what the user desired.



z/OS provides some extensions to the CF structure capabilities, specifically a way to override XES choices for structure placement.

ENFORCEORDER(YES) causes the system to enforce the order of CFs in the preference list in the process of structure allocation. It is a way to override the built-in system algorithms used by XES to determine structure locations. It is mutually exclusive with EXCLLIST. It may be the best and easiest way to put structures in the desired CF, for example, to take advantage of multiple CPs on one CF.



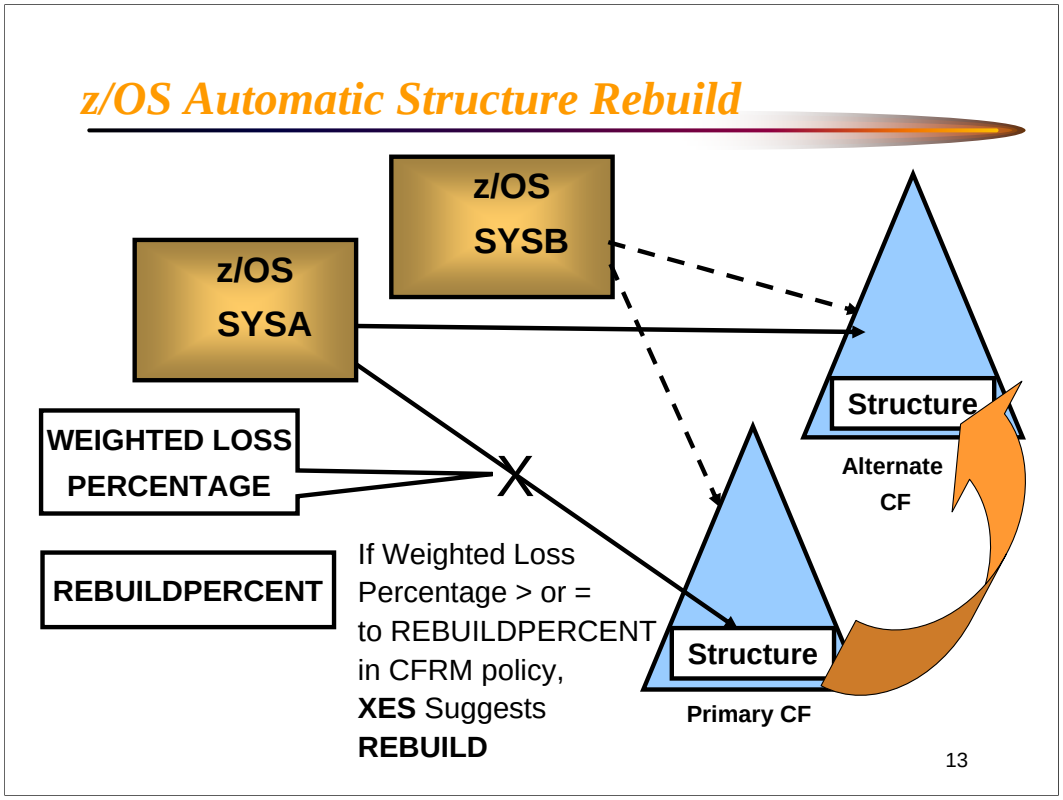
REBUILDPERCENT - This is used in recovery situations, and should always be 1.

DUPLEXED - Sets the DUPLEXing attribute

ALLOWED - Duplexing can be started with a command

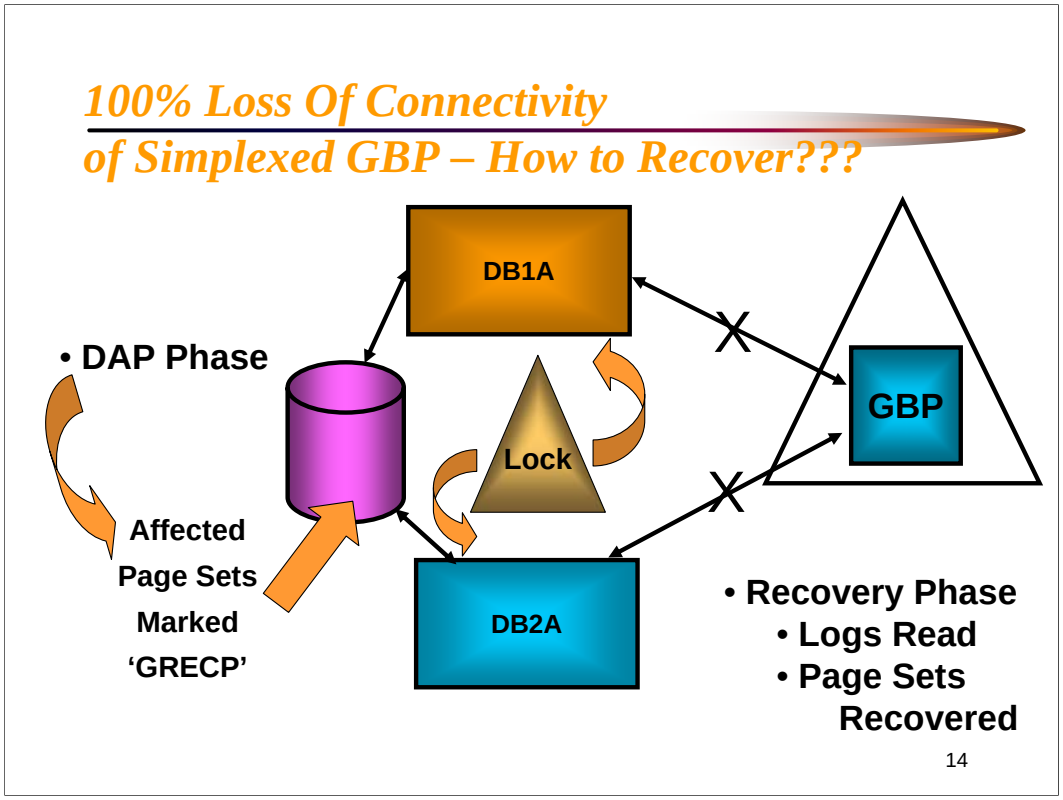
DISABLED - Duplexing turned off; run in simplex mode

ENABLED - Run with duplexed GBPs



z/OS supports the concept of *automatically rebuilding* structures. If the weight loss of connectivity (based on SFM weights for the z/OS images), is equal to or greater than the REBUILDPERCENT specified for the structure in the CFRM policy, then z/OS will signal the exploiter (e.g., DB2 or the IRLM) to REBUILD the structure in the alternate CF specified in the CFRM policy definition. To reiterate the requirements for structure rebuild:

- SFM policy in effect (or APAR OW30814 installed)
- Low REBUILDPERCENT specified in CFRM policy ('1' recommended)
- Alternate CF specified in CFRM policy
- Space available in alternate CF
- Connectivity to alternate CF



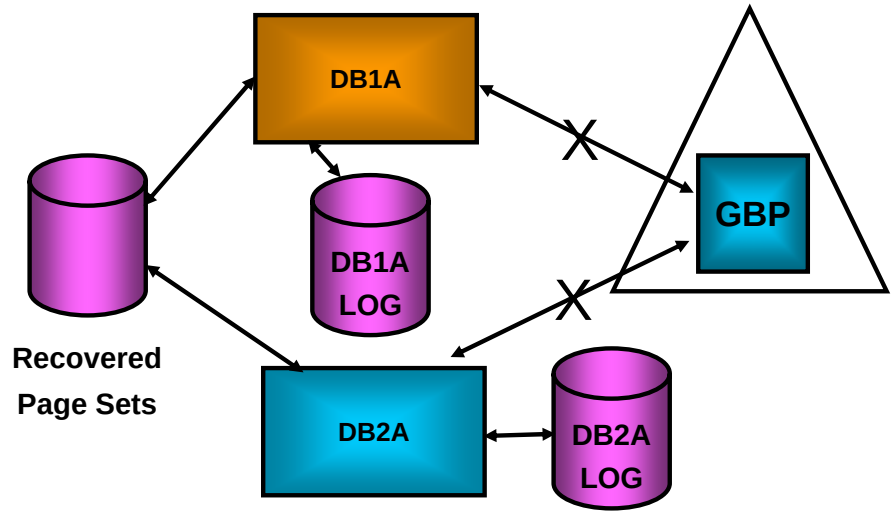
If a DB2 data sharing group loses 100% of connectivity to a group buffer pool, it cannot rebuild the GBP, since there is no remaining connection to the GBP. (Remember, a GBP 'rebuild' is really a COPY operation, requiring at least one DB2 to have access to the old GBP). A highly advertised feature of Version 5 of DB2 was the capability to automatically 'recover' page sets in the event of 100% loss of connectivity.

Before the DB2 group can do the recovery using the members' logs, it must identify the affected page sets that need recovery. DB2 must answer the question, "Which page sets had updated pages in the GBP at the time of total loss of connectivity?"

At first, this *damage assessment process* (DAP) sounds like an impossible task, but DB2, by examining the page set P-locks, and by other information kept in virtual memory about the open page sets, can, in fact, identify the affected page sets. They are marked with a new recover pending status, GRECP (Group Buffer Pool RECover Pending).

As the next visual illustrates, DB2 can then initiate an internal RECOVER process, to recover the affected page sets.

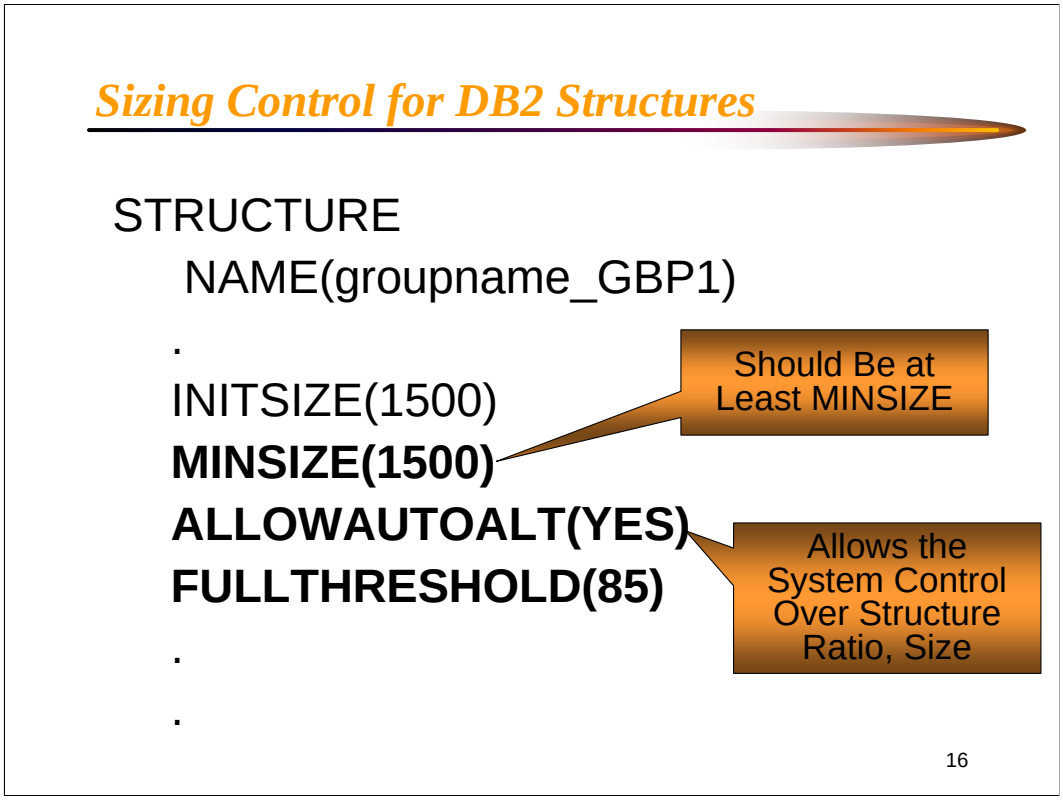
Simplexed GBP Page Set Recovery – How Long?



15

If the GBP was defined with AUTOREC(YES), DB2 does an internal recovery of the GRECP page sets, accessing the logs of members that updated the affected page sets. If AUTOREC is set to NO for the GBP, then manual START DATABASE commands must be issued to recover the affected page sets.

NOTE: LPL entries may have also been generated at the time of the loss, and may require manual START DATABASE commands to correct, especially in Version 7. Version 8 automatically initiates more recoveries of LPLs.

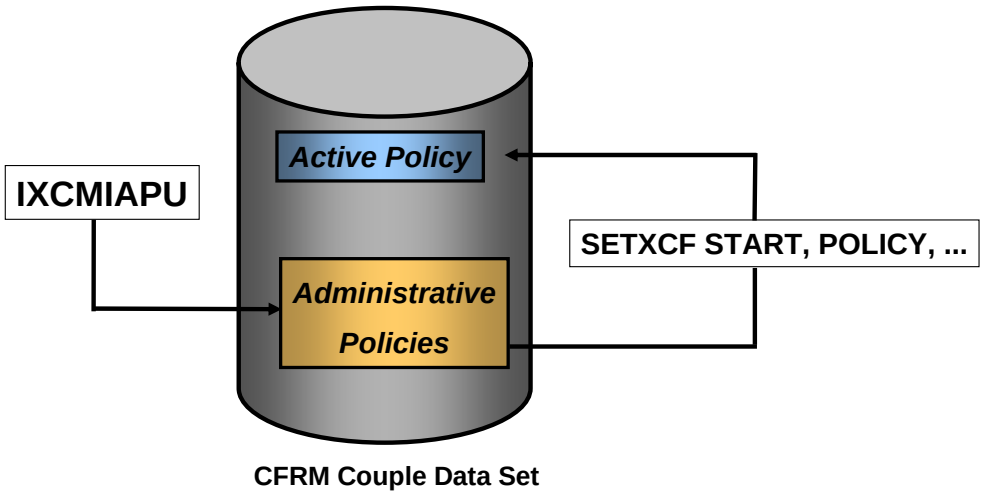


The ALLOWAUTOALT specification tells the z/OS system to monitor the size of the structure in question – in this example, it is a group buffer pool. When the structure is 'full', i.e., it is up to the percentage stated in the FULLTHRESHOLD field, z/OS XES, in conjunction with CFCC code, will automatically alter either the RATIO, or the INITSIZE of the structure, in order to alleviate the storage shortage.

In the case of group buffer pools, the system will first attempt to solve the storage problem by changing the RATIO. If that does not solve the shortage problem, then the size will be altered.

By specifying the MINSIZE parameter at least as high as the INITSIZE parameter, the system is instructed to not take any storage away from this structure that would take it below the INITSIZE, in the case where it was looking for storage for another structure that is short on storage.

ACTIVATING a CFRM Policy



17

When **IXCMIAPU** processes a set of definition statements for a policy, it stores the output in the administrative section of the CFRM couple data set. It does this so as not to destroy the active policy. The new definitions are made effective by issuing a manual command:

SETXCF START,POLICY,TYPE=CFRM,POLNAME=(xxxx)

The policy named in the SETXCF START command then takes the place of the currently active policy.

D XCF,STR; Displaying CF Structures

```
IXC359I          13.52.26          DISPLAY XCF
STRNAME          ALLOCATION TIME    STATUS
DSNDB0A_GBP0     06/08/2006 15:12:41   ALLOCATED
DSNDB0A_GBP1     06/08/2006 15:13:10   ALLOCATED
DSNDB0A_LOCK1    06/08/2006 15:00:53   ALLOCATED
DSNDB0A_SCA      06/08/2006 15:00:23   ALLOCATED
DSNDB0B_GBP0                           NOT ALLOCATED
DSNDB0B_LOCK1                          NOT ALLOCATED
DSNDB0B_SCA                            NOT ALLOCATED
ISTGENERIC       06/08/2006 13:14:47   POLICY CHANGE PENDING
```

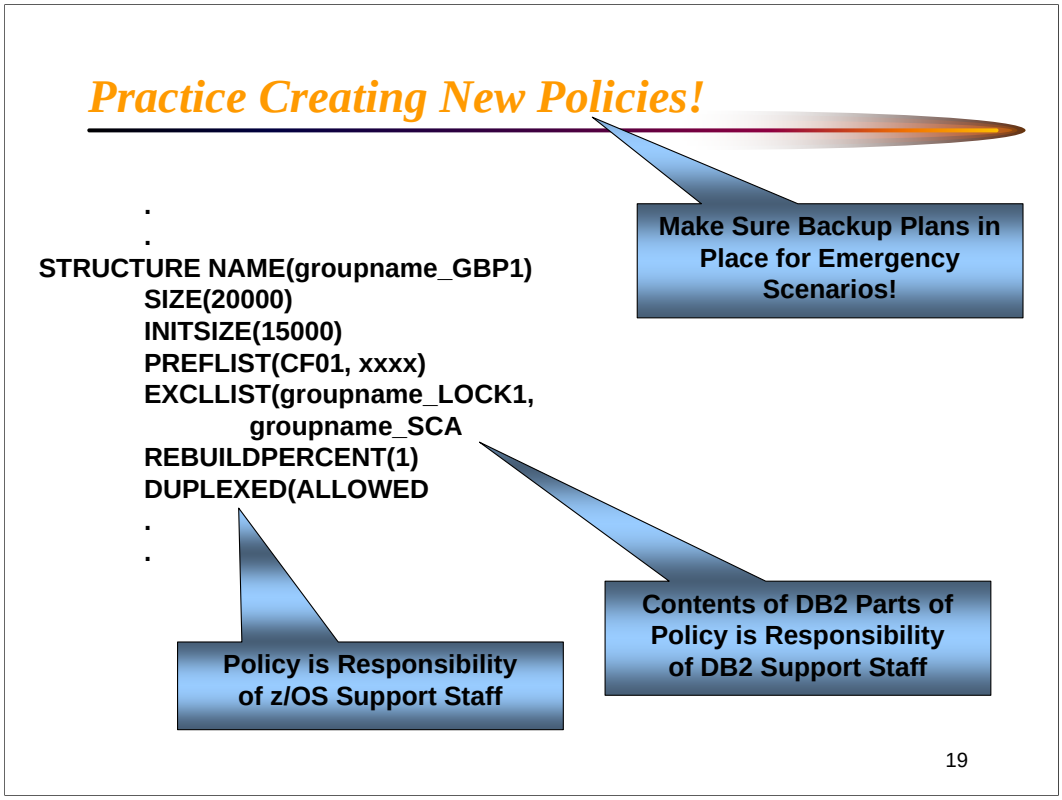
NOT An Error!
Structure Needs to be Rebuilt

18

The D XCF,STR command shows status information about the structures currently defined in the *active* CFRM policy. If specified without further qualification, summary information will be displayed about all coupling facility structures that are defined in the current active policy.

Note that a coupling facility may contain structures for multiple data sharing groups, as well as other z/OS subsystems. This example shows group buffer pools, (*_GBP0 and *_GBP1), lock structures, (*_LOCK1), and SCA structures, (*_SCA), for two data sharing groups, DSNDB0A and DSNDB0B. Also shown are structures used by other z/OS systems - ISTGENERIC used by VTAM, and two IXC * structures, used by XCF signaling services. These structures are extremely important to data sharing, as they support XCF communication services, which are heavily used by DB2 and IRLM.

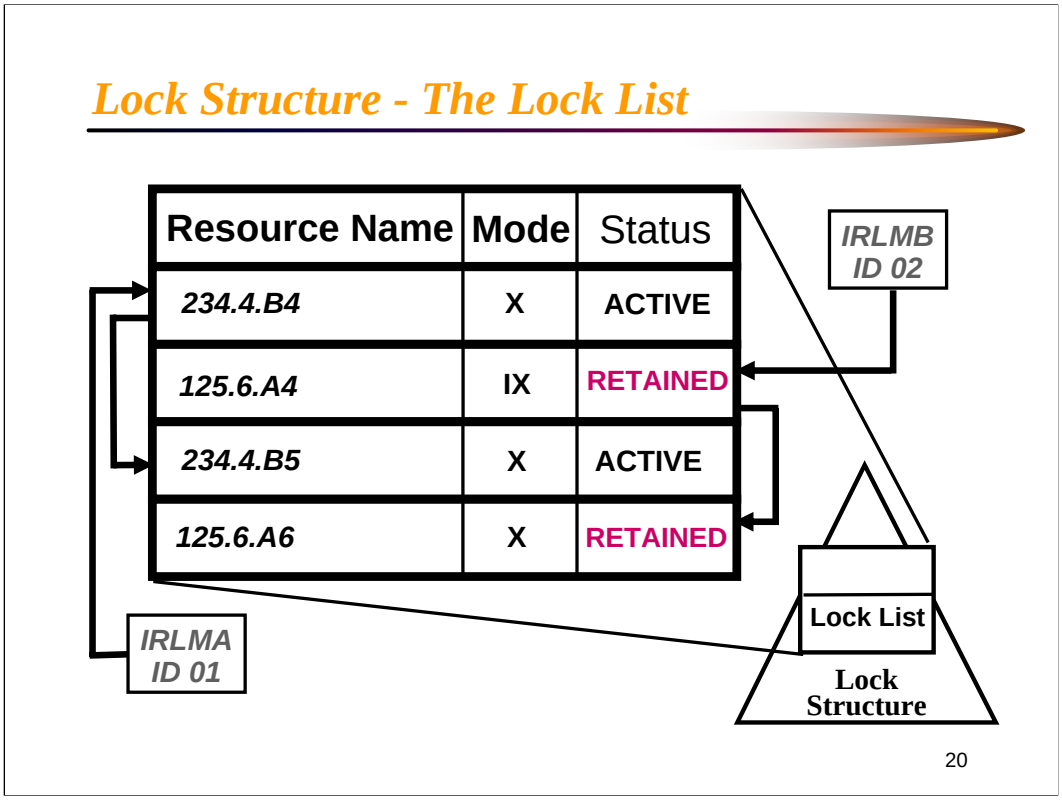
Note the status for the ISTGENERIC structure (used by VTAM). *POLICY CHANGE PENDING* means that a new CFRM policy was activated while this structure was allocated, and its definition changed. XES will not automatically rebuild the structure to the new specifications.



One tip, especially for new data sharing installations, is to make sure the procedure of creating a new policy is clearly understood, and practiced. The policy itself is typically the responsibility of the z/OS support staff, while the actual numbers for the DB2 structures are decided by the DB2 support staff.

When a policy is created and activated, and contains changed definitions, the status of the changed structure shows as 'POLICY CHANGE PENDING'. This just means what it says – there is a change in the definition of an allocated structure, and the system is waiting for the user to rebuild it. (z/OS does not automatically rebuild structures to incorporate definition changes).

Another important point regarding policies – make sure the installation has a backup plan for creating a new policy. Is a procedure in place for creating and activating a policy with changes needed quickly in an emergency???



For review purposes, let's take a look at the Lock List, or MRL (Modified Resource List), which consists of many entries. It is used by data sharing to track ALL modify locks (update locks) in the data sharing group – modify locks on both GBPD (Group Buffer Pool Dependent) page sets and non-GBPD page sets. Every time a DB2 system asks for any level of modify lock, it is recorded in this table in the lock structure.

The lock information for each lock includes DB2's internal designation for the resource, the mode of the lock, and the status, whether or not the lock is **ACTIVE** or **RETAINED**. A status of active means that the DB2 system is running. Retained, on the other hand, means that a DB2 and/or its associated IRLM has failed. In order to preserve data integrity, each of those locks protecting uncommitted data must be preserved. As illustrated in the diagram, the lock entries are queued by IRLM ID. In the case of an IRLM failure, the remaining IRLMs change the status of those locks of the failed IRLM to RETAINED. What is the impact of a retained lock? The next chart lists some of the results of a retained lock.

Type of Modify Locks That are Retained

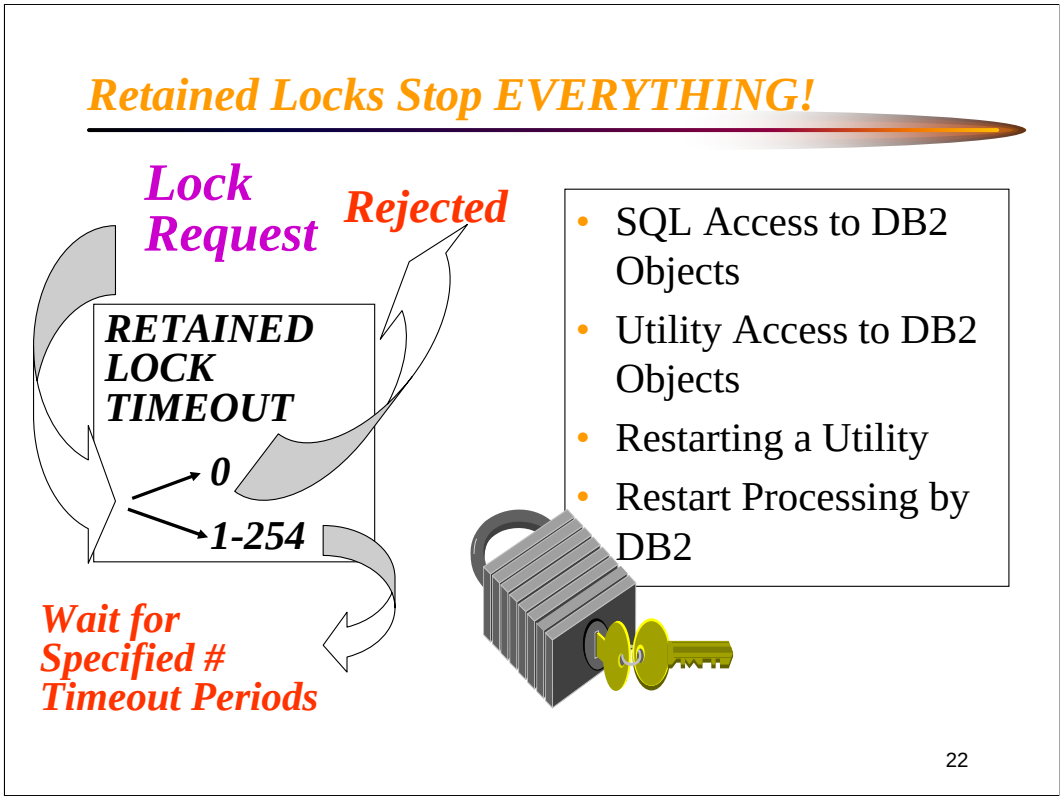
- L-Locks
  - Table Space Intent Locks
  - Table Locks
  - Page Locks
  - Row Locks
- P-Locks
  - Page Set P-Locks
  - Page P-locks
- Utility ID Locks

21

It is important to remember that ALL modify locks are retained in the event of an abnormal DB2 termination. This includes transaction, or L-locks, on DB2 objects. These locks must be retained to maintain data integrity.

P-locks, both page set and page P-locks are also retained. If they are exclusive in nature, they will stop all access attempts against the objects on which the terminated DB2 had them.

A retained utility ID lock will prevent that utility from being restarted.

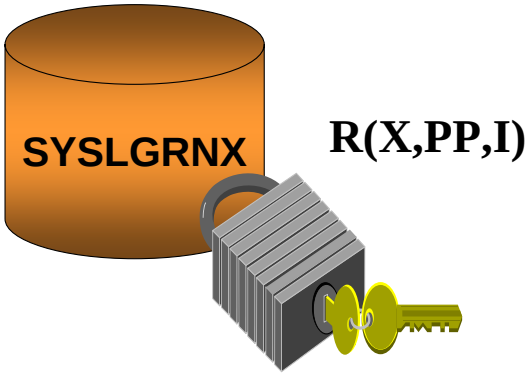


Now that we understand what locks are retained in the event of a DB2 failure, we must consider the consequences of retained locks. In general, they STOP EVERYTHING! A new request with an incompatible state will be rejected.

The exception to this rule depends on the setting of the RETAINED LOCK TIMEOUT value in ZPARM (set at installation time on panel DSNTIPI). If it is set at 0, DB2 immediately rejects the new incompatible lock request. If it is a non-zero number, DB2 and the IRLM will wait for the specified number of IRLM timeout periods before rejecting the new request.

The ONLY way to clear retained locks is by restarting the failed DB2 subsystem.

A Retained Lock's Effect



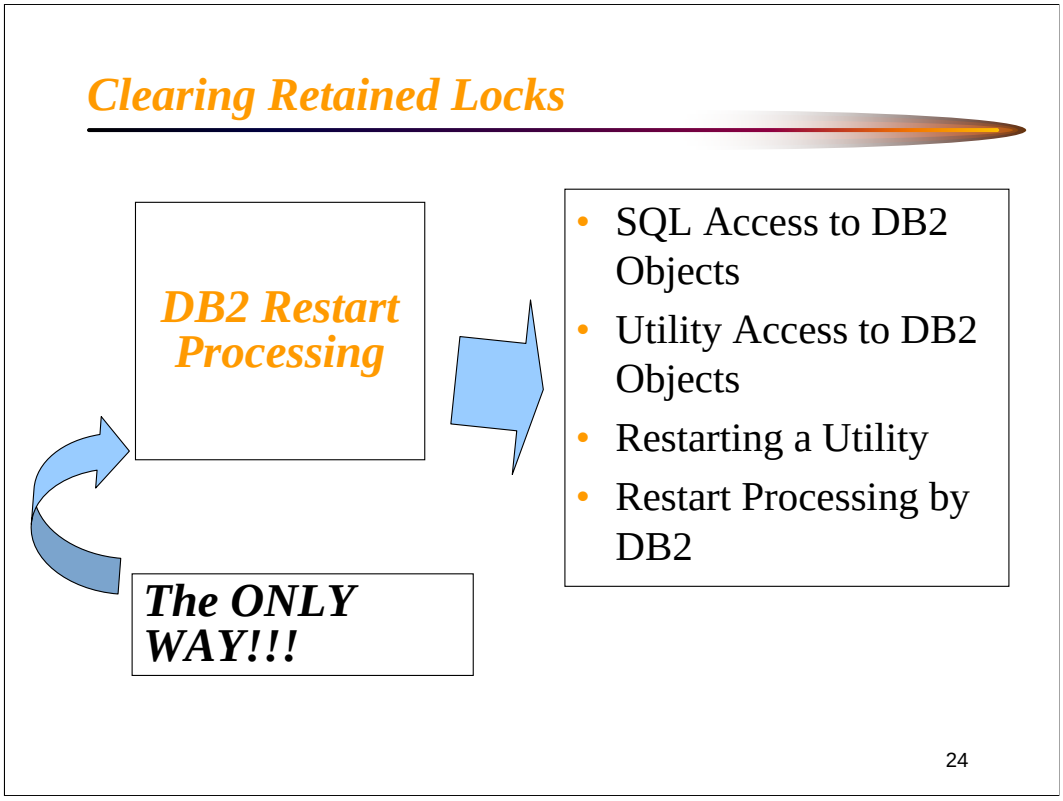
23

One of the consequences of retained locks, as we have already seen, is that, in general, they STOP EVERYTHING as far as access to the locked object! But the effect on a DB2 subsystem may be much more serious than that.

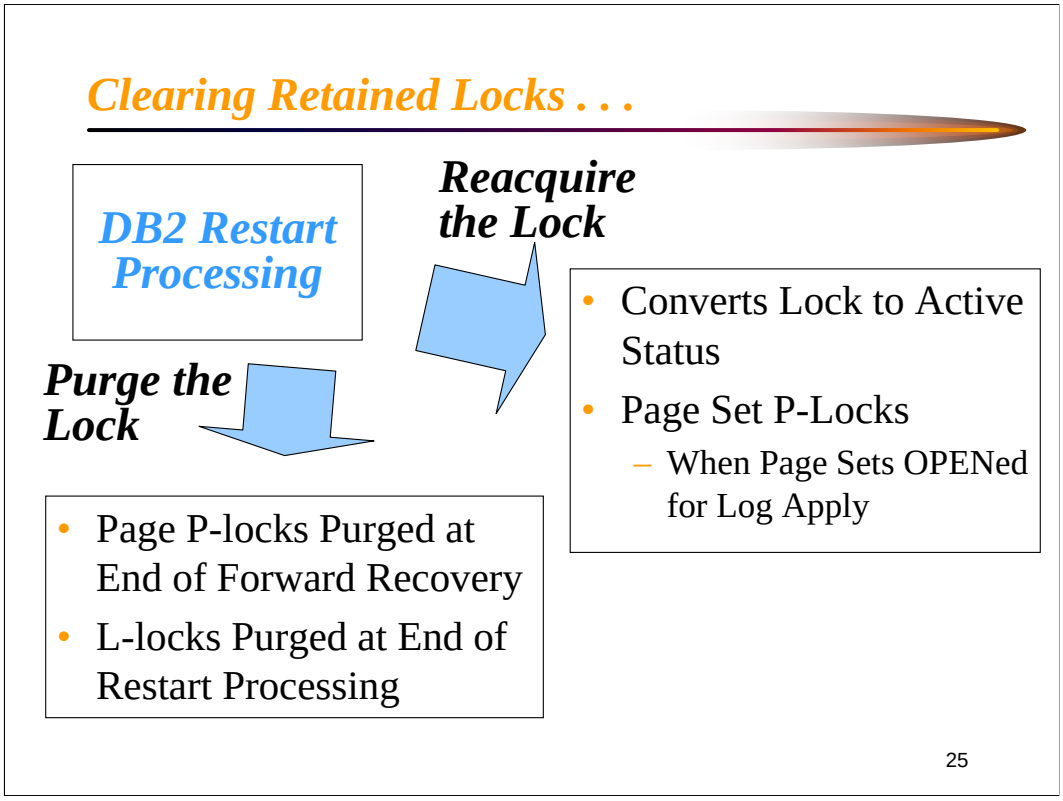
For example, if the lock illustrated in the chart above were to happen (i.e., a DB2 or IRLM would abend holding an exclusive lock on SYSLGRNX), what would the effect be?

If this were a data sharing group, and no member in the group could access SYSLGRNX, no updates could happen! This would guarantee lots of phones ringing!

An interesting question to ponder: Why should the illustrated lock never happen??



Restarting a failed DB2 subsystem is the ONLY way to clear retained locks and preserve data integrity. The next chart goes into a bit more detail on how DB2 restart processing clears the retained locks which are protecting uncommitted resources.



The DB2 restart process clears retained locks in one of two ways. The first is to convert the lock back to an active status, called *reacquiring* the lock. This is what DB2 does for page set P-locks. Once reacquired, the page set P-locks are available for any necessary negotiation during restart processing.

The second method of clearing a retained lock is to *purge* it. DB2 does this for L-locks, after restart has finished processing on the object – i.e., either finished committing the work on the object, or rolling it back. Page P-locks are purged at the end of forward recovery, as the URs that may have been accessing pages at the row level have been processed, and the global latch capability provided by the page P-lock is no longer needed.

Displaying Retained Locks with a z/OS Command: F irlmproc,STATUS,ALLD - IRLM Status

F IRA1IRLM,STATUS,ALLD

DXR102I IRA1 STATUS IRLMID=001

SUBSYSTEMS		IDENTIFIED		PT01
NAME	STATUS	RET_LKS	IRLMID	RLM NAME
DB1A	UP	0	001	IR1A
DB2A	UP	0	001	IR2A

26

F (MODIFY) *irlmproc*,STATUS,ALLD is a z/OS command which enables you to see all DB2 subsystems in a data sharing group that are or have been connected to one IRLM group. The STATUS can be UP, DOWN, or SYSFAIL.

The most important information on the display is shown in the RET_LKS column. This number indicates the number of retained locks held by a subsystem that failed or was running on an IRLM that failed. If retained locks are shown, the DB2 that failed must be restarted to clear those retained locks.

This example shows two DB2s, DB1A and DB2A, both of which are in UP status, and therefore neither has any retained locks.

In the production DB2 world, of course, an abnormal DB2 termination would almost certainly produce some retained locks. But this command is a quick method of double-checking if retained locks do exist. It may be a good operational step to issue this command after a RESTART LIGHT of DB2, to see if any of the few retained locks that a light restart cannot clear still exist.

Sample ARM Policy

DATA TYPE(ARM)

DEFINE POLICY NAME(ARMPOL1)

RESTART GROUP . . .
- **TARGET SYSTEM . . .**
- **RESTART PACING . . .**
- **ELEMENT . . .**
- **RESTART_ATTEMPTS (max, interval)**
- **TERMTYPE …..**
- **RESTART METHOD (. . .**

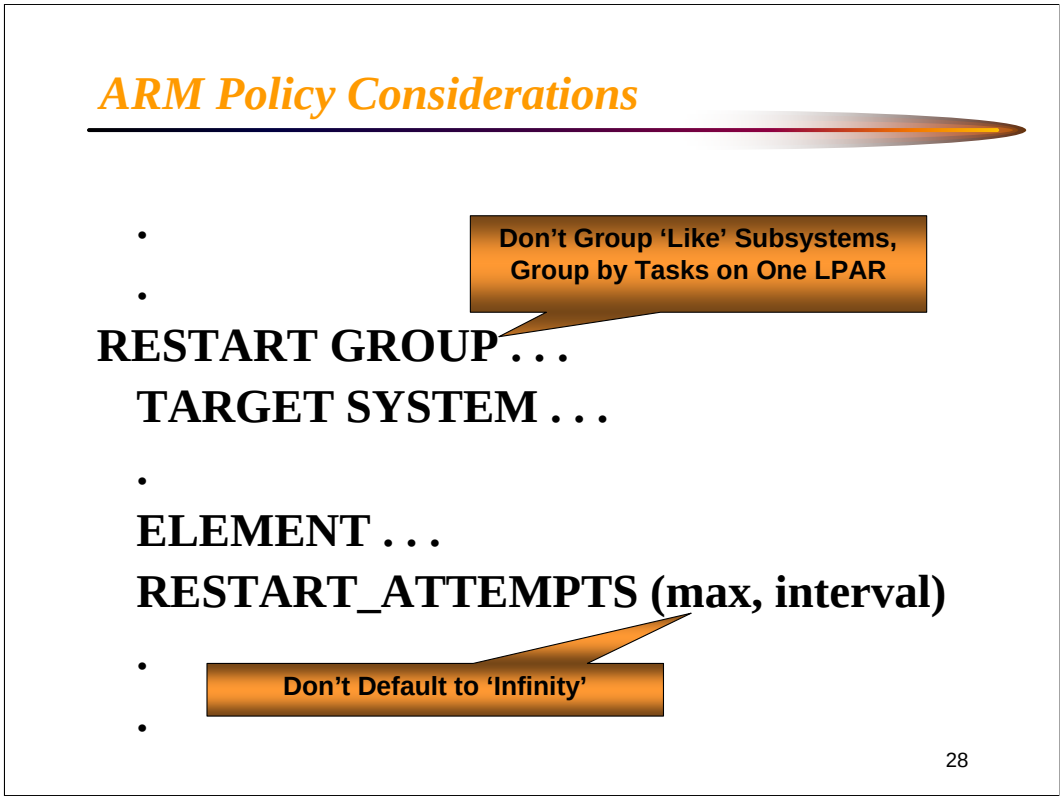
27

The ARM (Automatic Restart Management) capabilities of z/OS are very powerful. To fully explore the possible parameters is far beyond the intent of this presentation.

The above skeleton example of an ARM policy illustrates some of the main points of ARM. Subsystems, called ELEMENTS, can be GROUPed together, and restarted together, e.g., DB2, IRLM and CICS.

Restart attempts allows the user to limit the number of times the system will attempt to restart a subsystem.

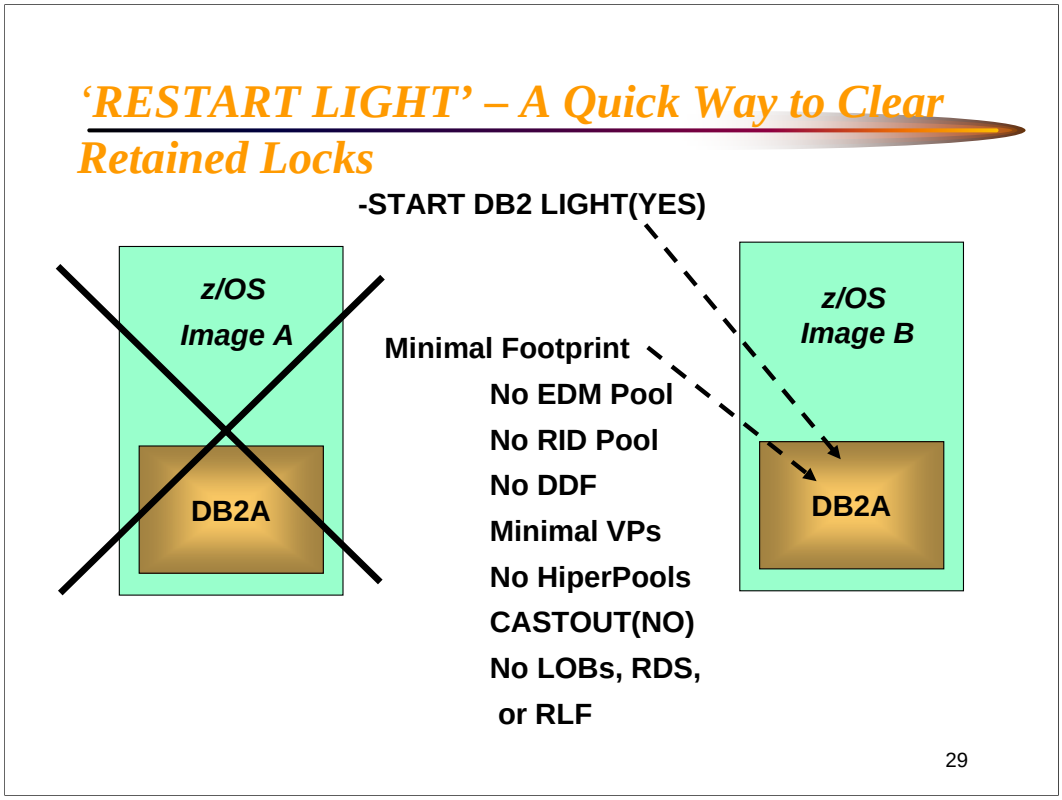
For a detailed list of considerations in using ARM, see *MVS: Setting up a Sysplex.*



When specifying the groups for ARM to restart, don't be tempted to put 'like' systems together. One installation did that, and then, when one LPAR failed because of an unrelated problem, ARM started the DB2 subsystem on the second LPAR, the related CICS on the third, and the related MQ on the fourth. In the customers own words, the subsystems started to scream, 'where is my buddy?'

Restart attempts allows the user to limit the number of times the system will attempt to restart a subsystem. DO NOT default this value to 'infinity'. If there is something wrong with the subsystem such that it cannot start, ARM will keep trying to restart it. The restart messages will flood z/OS message buffers, and the z/OS staff will not appreciate it!

For a detailed list of considerations in using ARM, see *MVS: Setting up a Sysplex.*

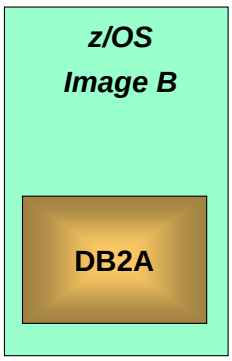


As we have seen, retained locks protect objects with uncommitted URs that were interrupted by an abnormal DB2 failure. In order to make those objects available to the rest of the group, that member must be restarted to clear the retained locks. Since it may not be possible to restart the DB2 on another z/OS image due to system resource constraints, DB2 Version 7 introduced the concept of RESTART LIGHT. RESTART LIGHT is a fast, minimal resource kind of restart for data sharing. As noted on the chart, the 'footprint' of a RESTART LIGHT is very minimal.

RESTART LIGHT will enable DB2 to restart more quickly, clear the retained locks, and then terminate. This new restart would be ideal to team with an automatic restart automation software, such as ARM (Automatic Restart Manager).

'RESTART LIGHT' – Further Considerations

DSNY009I @DSNYSTRT SUBSYSTEM STARTING IN LIGHT MODE, NORMAL TERMINATION TO FOLLOW RELEASE OF RETAINED LOCKS



Retained Locks Released EXCEPT:

- **Indoubt Units of Recovery (V7)**
- **Postponed-Abort Units of Recovery**
- **IX Mode Page Set P-locks (V7)**

30

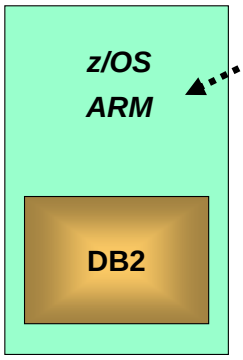
As noted in the DSNY009I message, DB2 will normally terminate after a light start.

The primary objective of a light restart is the release of retained locks that the DB2 subsystem was holding at the time of abnormal termination. However, as noted on the chart, some retained locks cannot be released by a light restart (esp. a V 7 restart) – those held by indoubt URs, postponed-abort URs, and IX mode page set P-locks.

A retained IX mode page set P-lock will not prevent access to the page set by other applications; however, it will block access by drainers, such as utilities.

'RESTART LIGHT' – ARM Considerations

RESTART_METHOD(SYSTERM,STC,'cmdprfx STA DB2,LIGHT(YES)')



IRLM Specification should include PC=YES:
RESTART_METHOD (SYSTERM,STC,
'cmdprfx S irlmproc,PC=YES)')

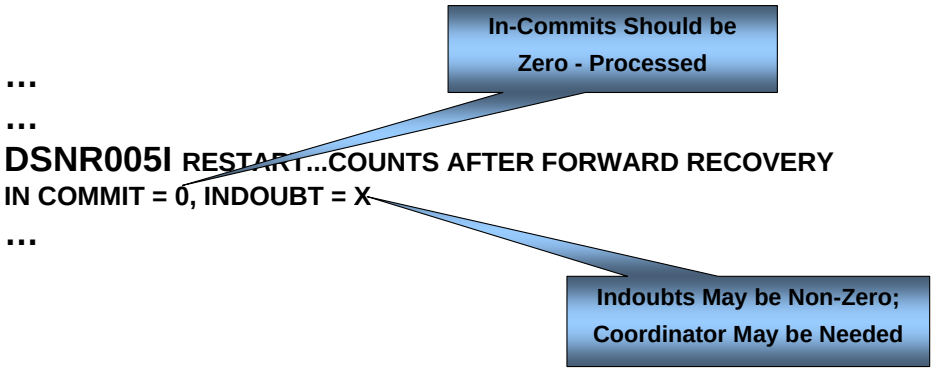
31

A LIGHT start can be specified in an ARM policy. This will enable the system to automatically restart a failed DB2 as quickly as possible, so that data accessibility problems are not caused by retained locks.

The ARM policy specification for both DB2 and IRLM are shown in the chart. Note that the specification of PC=YES will allow the IRLM to obtain the full benefits of restart-light mode by having its locks in extended private storage instead of ECSA (Extended Common Storage Area).

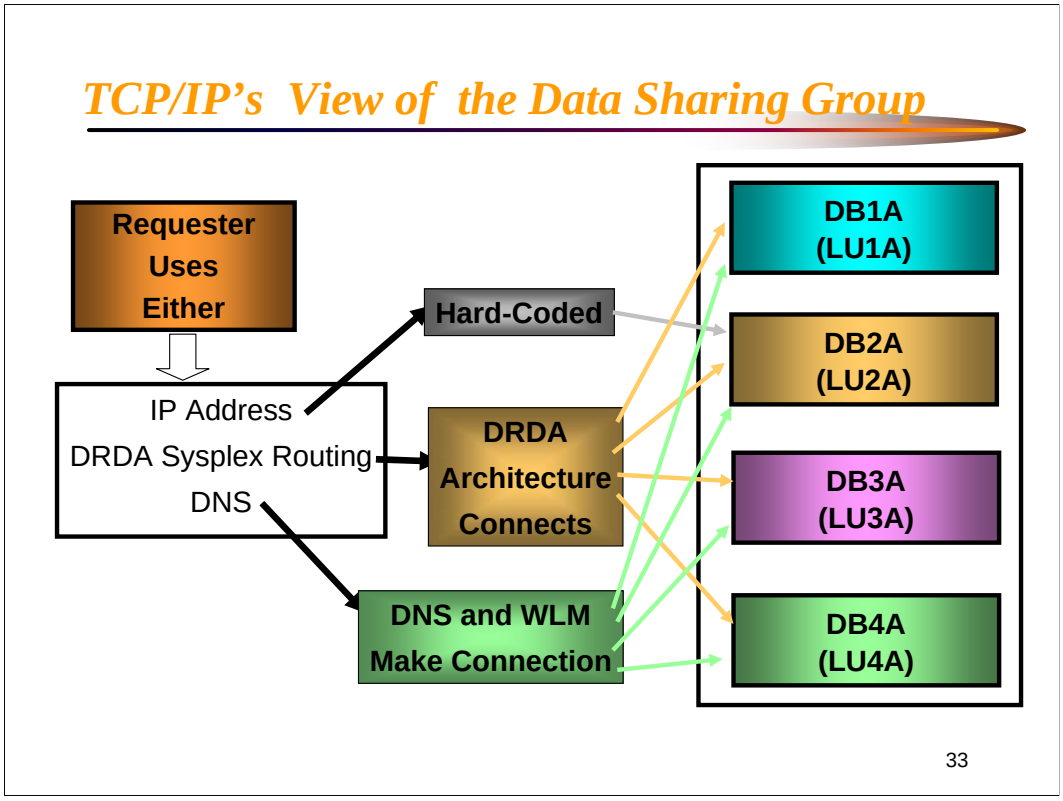
ARM will not restart the member again after the light restart has completed successfully.

DB2 Restart Forward Log Recovery (FLR) Messages



32

DSNR005I: This message reports the counts of in-commit and indoubt URs. In-commits should have been processed successfully, and the count should be zero. Some indoubt URs may still remain, however. (Communication with another system may be needed to properly process an indoubt UR). **NOTE**: Version 7 RESTART LIGHT will not accept connections from the other systems involved, and indoubt threads are not processed. Version 8 RESTART LIGHT does attempt to re-establish communications with the two-phase commit partner, and attempts to clear up the indoubt threads.



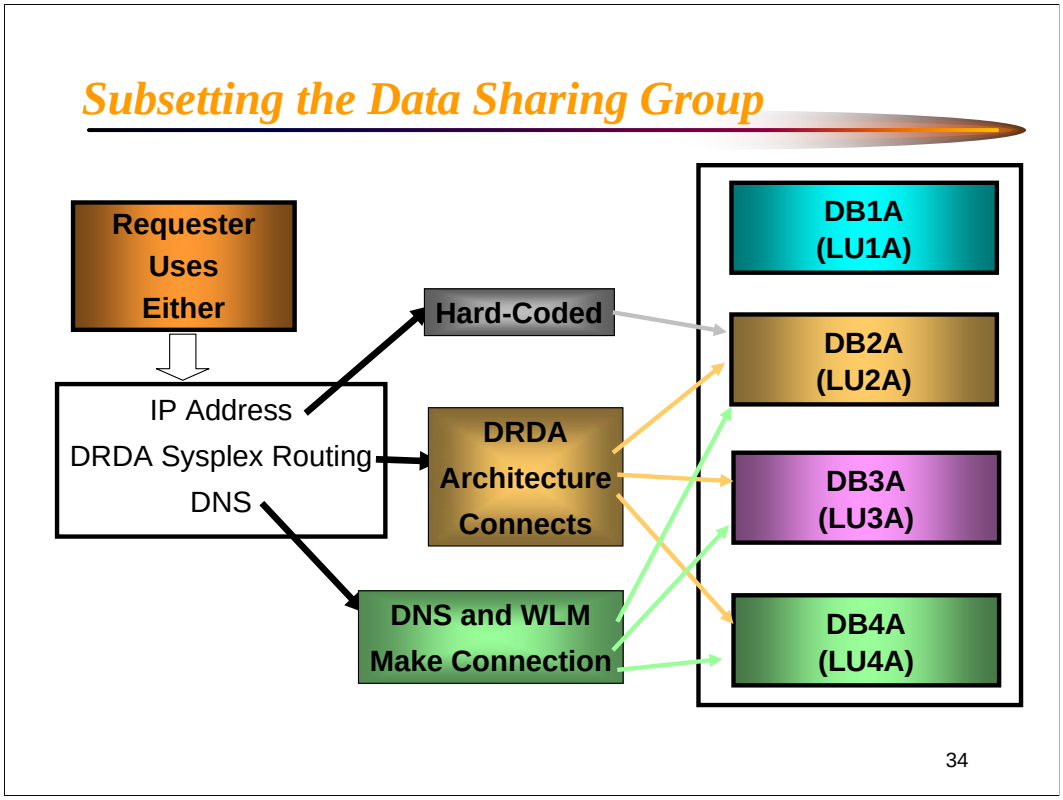
A requester connecting to a DB2 data sharing server using TCP/IP must use DRDA protocols, and must support DRDA level 3.

The requester may enter the desired DB2 member's IP address in its TCP/IP configuration - the 'hard-coded' method. Any workload balancing is a very labor-intensive manual process, and if a DB2 member is restarted on another CPC, its IP address will change.

By correctly setting up the client configuration, a requester can take advantage of DRDA architecture, and the workload balancing information 'built in'. See the supplied documentation on the client for the details. If you cannot use DRDA workload balancing, use the DNS (Domain Name Server) to balance the workload.

To use the DNS approach, also called the connection optimization for a Sysplex, the following steps must be taken:

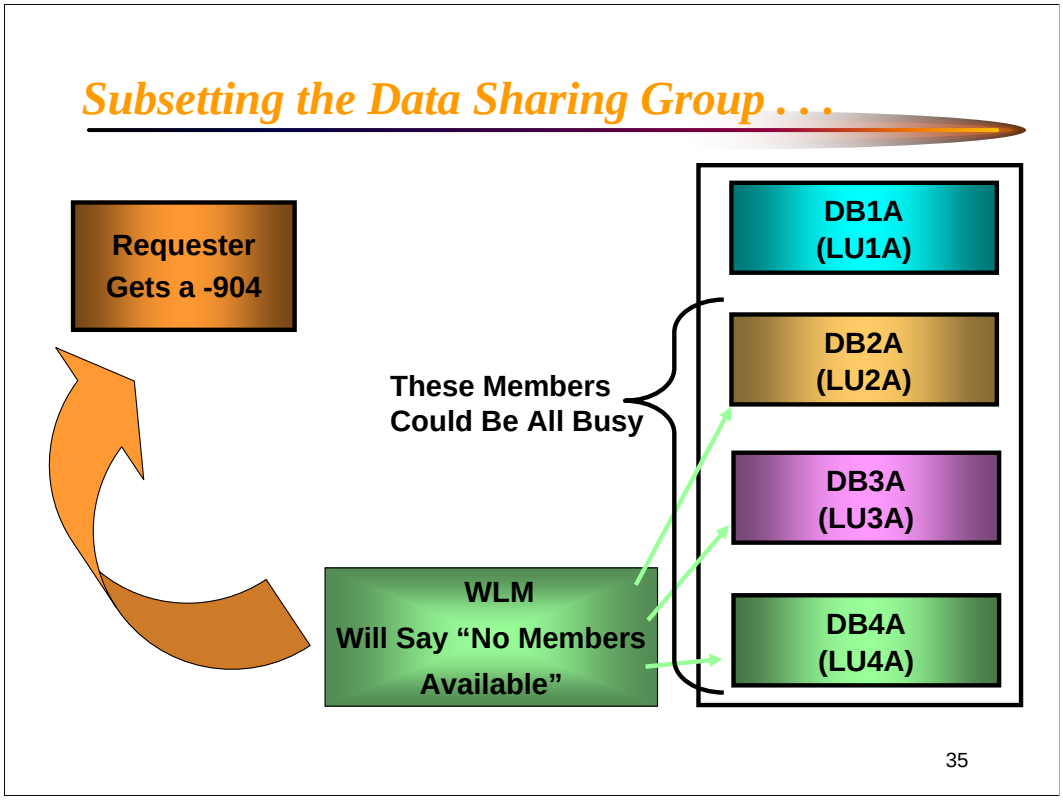
1. Convert all members in the group to goal mode.
2. Set up the DNS with a bootfile entry for the sysplex.
3. WLM then registers the group and server names to DNS.



Transparently to end users, the members of a data sharing server group may be ‘subsetted’. To exclude a member from DDF processing, set the MAX REMOTE ACTIVE option on installation panel DSNTIPE to zero.

This has the effect of not being included in WLM’s list of potential systems to connect to.

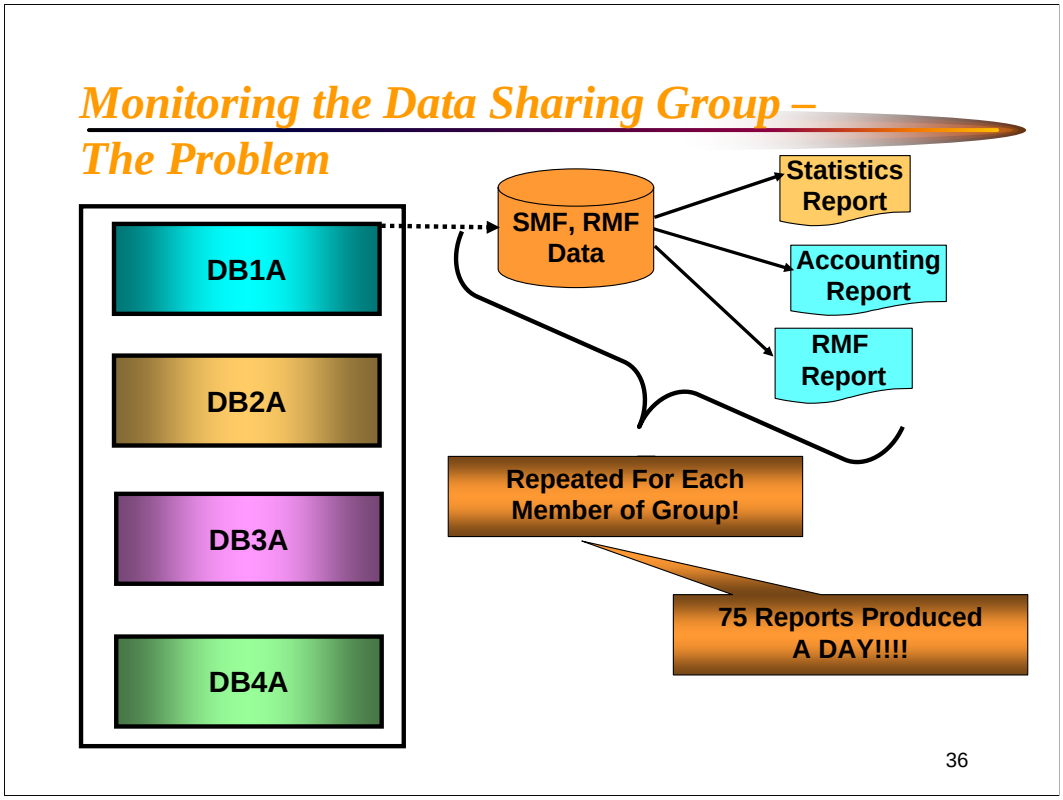
A potential problem with this approach is shown on the next chart.



The potential problem with the approach of subsetting the group is that WLM may determine that the eligible members are all too busy to accept to work. Thus, the list of potential members to connect to would be passed back without any members in it – the requester would get a -904 0D31038 Resource Type 00001004.

This can be very frustrating to find, as it will happen intermittently.

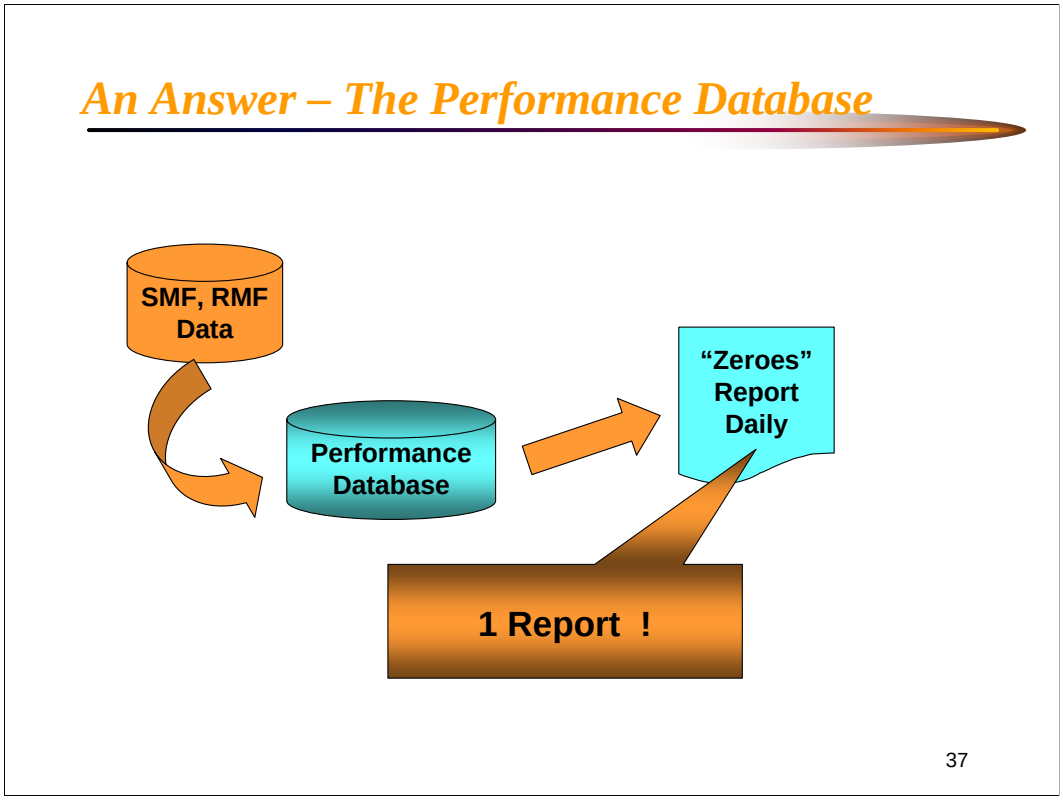
If you decide to subset the data sharing server group, be aware of this potential problem.



One of the problems encountered when monitoring a data sharing group is not the lack of data, but far too much of it. For example, the SMF record volume by itself may be a problem. Beyond that, the number of reports produced on a daily basis can be overwhelming – one large data sharing account produces about 75 reports a day.

Question: Who is going to look at them??

There has got to be a better way – the next few charts introduce a way to handle this massive amount of data.

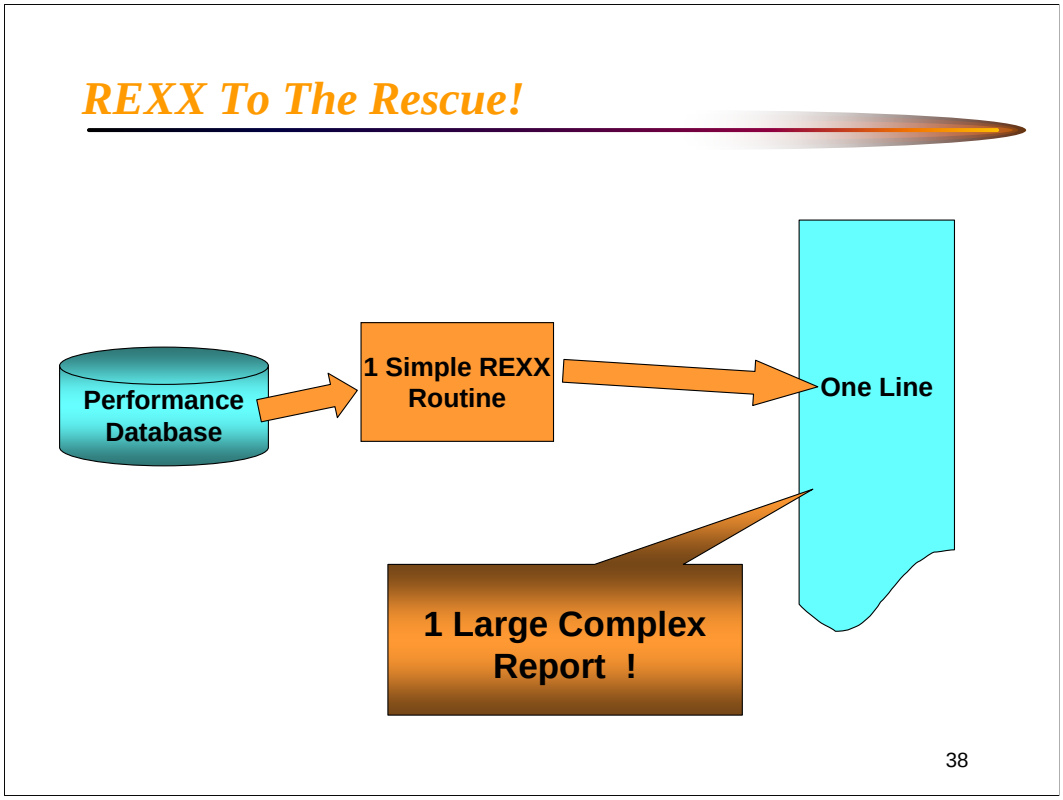


One possible solution to the problem of too much data is the performance database. There are multiple vendor solutions here – it does not have to be 'home-grown'.

Once the performance database is built, and the overhead necessary to do it has been planned for, the next step is just as important. There needs to be a way to 'boil down' the still massive data available in the performance database.

One way to do that is a daily 'zeroes' report – a summarization of the performance database. SMF counters that are important to the account should show up on the report, and the ones that should be zero will stand out very prominently if they are non-zero.

Of course, putting all the important SMF counters on one report could involve a very complex program. But there is a simpler way to do that – shown on the next chart.



Once the idea of an overall ‘zeroes’ report is implemented, the maintenance of the program becomes a concern - it could get very complex, very quickly. Once the performance database and its contents become know to the installation’s departments, everyone will find needed data there somewhere.

One approach to this complexity problem is to utilize REXX – one simple REXX routine, with perhaps 1 or 2 SQL statements, to produce each line of the report. Debugging and maintenance are greatly simplified, and when a new line is requested for the report, only one simple routine needs to be written, and inserted into the string of REXX routines.

For an example of one installation’s zeroes report, please see the next few charts.

The "Zeroes Report" – Data Sharing Portion

```
.
.
.
*****_____*****
*****  DATA SHARING          *****
*****_____*****
CASTOUT ENGINE NOT AVAILABLE                              0
COUPLING FACILITY WRITE ENGINE NOT AVAILABLE              0
COUPLING FACILITY READ NOT COMPLETE                       0
COUPLING FACILITY WRITE NOT COMPLETE                      0
GROUP BUFFER POOL REBUILD BY THIS MEMBER                  0
EXPLICIT CROSS-INVALIDATIONS                              133263
PAGE P-LOCK SUSPENSIONS FOR SPACE MAP PAGES               11
PAGE P-LOCK SUSPENSIONS FOR DATA PAGES                    0
PAGE P-LOCK SUSPENSIONS FOR INDEX LEAF PAGES              0
.
.
.
```

39

This chart shows part of the currently implemented 'zeroes' report in one large financial services installation. The data sharing section is one of 12 sections currently part of the zeroes report.

The first few lines illustrate why this is called a 'zeroes' report. There are counters that should be zeroes, if everything is working correctly. For example, castout engines not available should ideally be zero – if they are not, it may be a sign of a system not tuned to properly handle the update workload.

CF Reads and Write failures should always be zero – if they are not, there is the possibility of unavailable data to the installation's customers.

Explicit cross-invalidations is a quick way to gauge how many LOBs are being processed in the installation.

Page P-lock suspensions is a good way to look for possible locking problems in data sharing's handling of sub-page concurrency scenarios – updates to space map pages, row-level locking in data pages, and index leaf page updates.

The "Zeroes Report" – Data Sharing Portion

.
.
.

PAGE P-LOCK NEGOTIATIONS FOR SPACE MAP PAGES	15
PAGE P-LOCK NEGOTIATIONS FOR DATA PAGES	0
PAGE P-LOCK NEGOTIATIONS FOR INDEX LEAF PAGES	0
SUSPENDS DUE TO IRLM GLOBAL RESOURCE CONTENTION	4140852
SUSPENDS DUE TO MVS XES GLOBAL CONTENTION	17766031
SUSPENDS DUE TO FALSE CONTENTION	2755017
ENGINE NOT AVAILABLE FOR P-LOCK OR NOTIFY EXIT	0
UNSUCCESSFUL AUTHORIZATION CHECKS	15
UNSUCCESSFUL CLAIM REQUESTS	794
UNSUCCESSFUL DRAIN REQUESTS	21

.
.
.

40

This continuation of the data sharing section of the report shows some potential areas for further checking – suspends are happening because of global IRLM and XES contentions.

If too many false contentions show on the report, it is a sure sign that the hash table, or lock table, portion of the lock structure is too small.

Remember, each line is produced by one simple REXX routine – making the implementation and maintenance of this report program a relatively simple programming exercise.

Summary - Getting Started and Monitoring Data Sharing

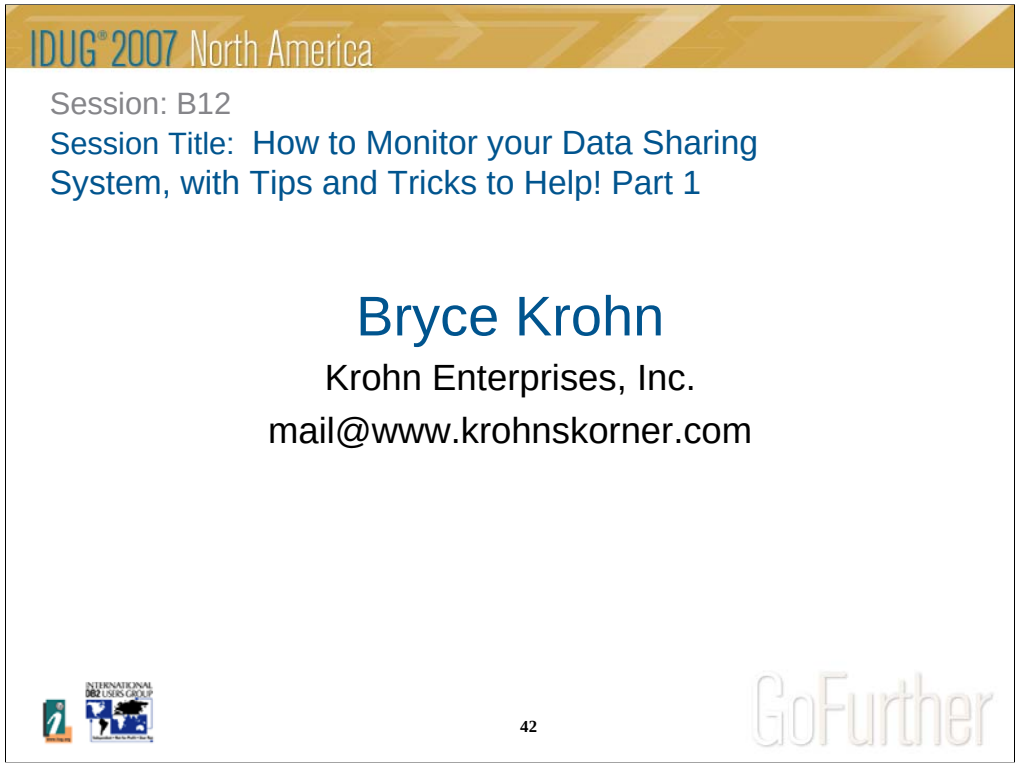
- **DB2 CF Structures Need to be Defined**
- **Sizes Need to be Estimated**
- **Suggestions in DB2 Manuals**
- **CF Sizer Available**
  - **Assumes Latest Hardware**
- **Policy Needs to be Initialized**
- **Data Sharing Needs To Be Monitored**

41

This presentation has discussed the estimation and planning for the size and placement of the DB2 structures, as well as provided checks for existing data sharing customers to check.

When the initial names, sizes, and placements have been decided, a CFRM (Coupling Facility Resource Manager) policy must be initialized and put in place in the z/OS system.

Once data sharing has been implemented, several tips have been covered in methods to do that monitoring.



Monitoring a data sharing system can be a daunting, time-consuming task. But there is a real need to keep track of how your data sharing system is running. This part of the presentation has discussed the major steps needed to get started in data sharing, as well as definitions that should be checked for existing data sharing customers. Several tips for monitoring have been included.