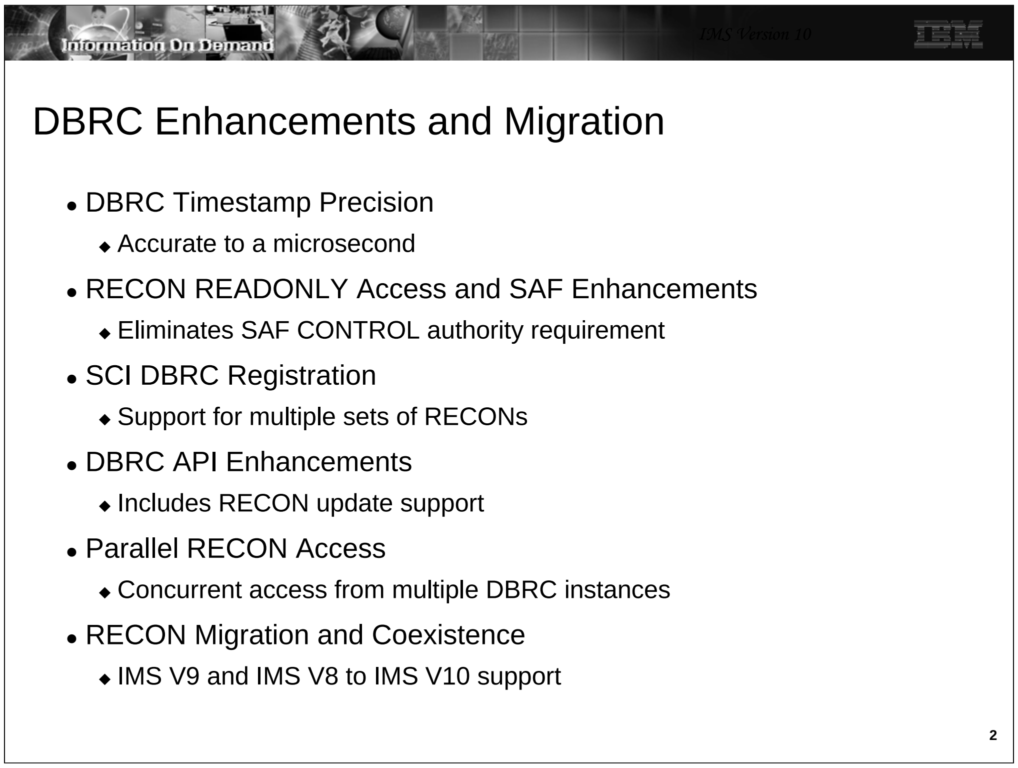
Information On Demand

IMS Version 10

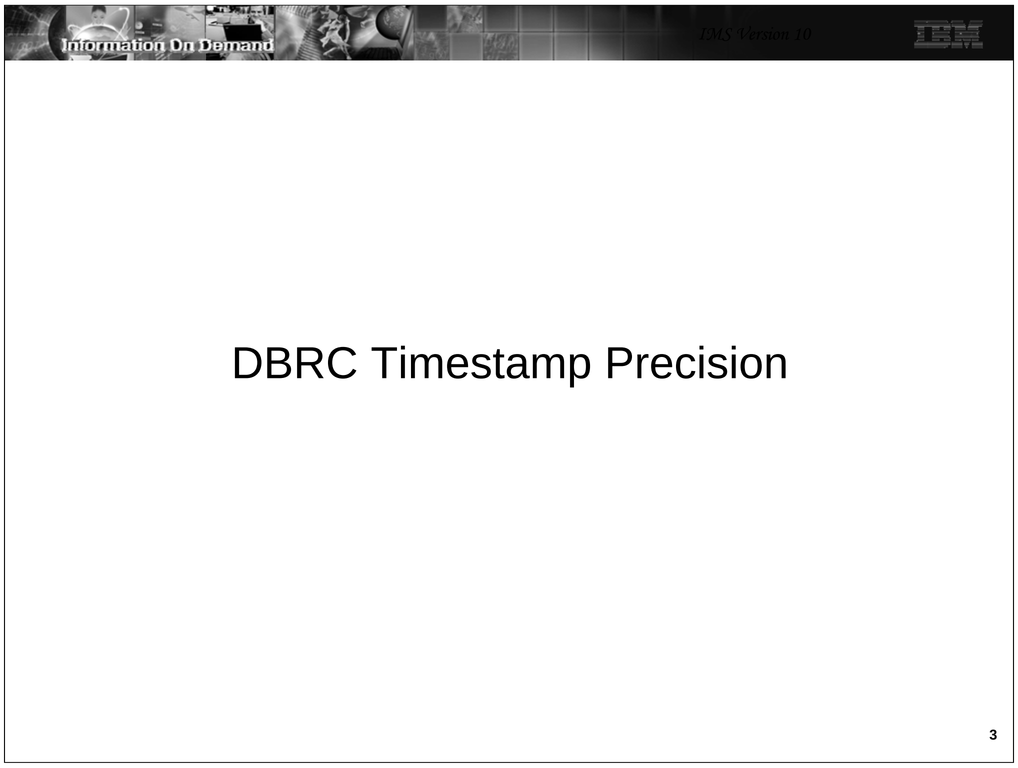
DBRC

ON DEMAND BUSINESS

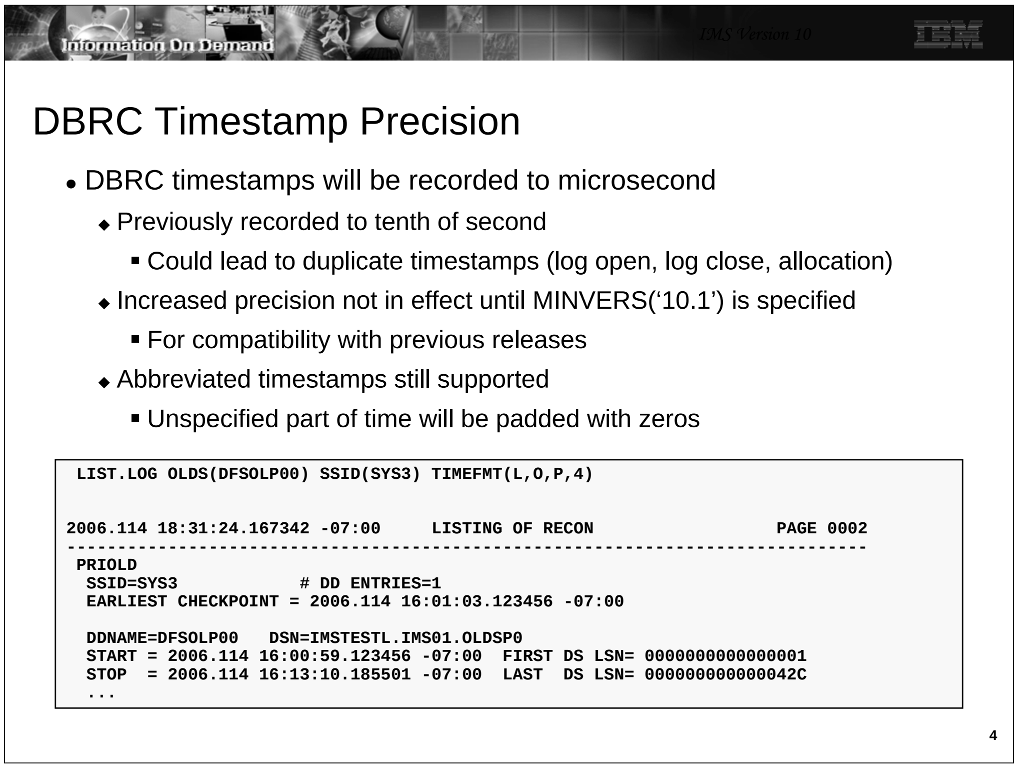


DBRC Enhancements and Migration

- DBRC Timestamp Precision
  - Accurate to a microsecond
- RECON READONLY Access and SAF Enhancements
  - Eliminates SAF CONTROL authority requirement
- SCI DBRC Registration
  - Support for multiple sets of RECONs
- DBRC API Enhancements
  - Includes RECON update support
- Parallel RECON Access
  - Concurrent access from multiple DBRC instances
- RECON Migration and Coexistence
  - IMS V9 and IMS V8 to IMS V10 support



DBRC Timestamp Precision



DBRC Timestamp Precision

- DBRC timestamps will be recorded to microsecond
  - Previously recorded to tenth of second
    - Could lead to duplicate timestamps (log open, log close, allocation)
  - Increased precision not in effect until MINVERS(‘10.1’) is specified
    - For compatibility with previous releases
  - Abbreviated timestamps still supported
    - Unspecified part of time will be padded with zeros

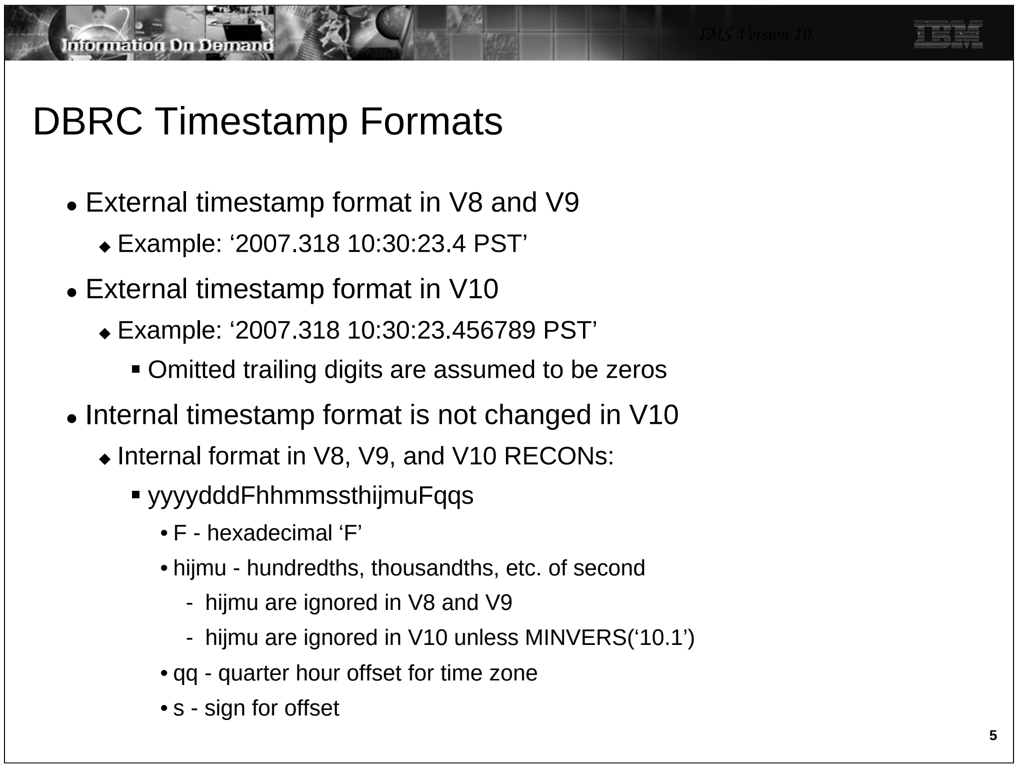
```
 LIST.LOG OLDS(DFSOLP00) SSID(SYS3) TIMEFMT(L,O,P,4)


2006.114 18:31:24.167342 -07:00     LISTING OF RECON                 PAGE 0002
-------------------------------------------------------------------------------
 PRIOLD
  SSID=SYS3            # DD ENTRIES=1
  EARLIEST CHECKPOINT = 2006.114 16:01:03.123456 -07:00

  DDNAME=DFSOLP00   DSN=IMSTESTL.IMS01.OLDSP0
  START = 2006.114 16:00:59.123456 -07:00  FIRST DS LSN= 0000000000000001
  STOP  = 2006.114 16:13:10.185501 -07:00  LAST  DS LSN= 000000000000042C
  ...
```

4

IMS V10 will record timestamps to the microsecond when MINVERS('10.1') is in effect. Previous releases record timestamps to the tenth of a second. When a user specifies a timestamp, it may be abbreviated. That means that times to the microsecond do not have to be specified. Unspecified parts of the time are padded with zeros.

When using GENJCL, the precision value may be coded on the TIMEFMT parameter of the %SET statement in skeletal JCL. It is a value from 1 to 6. The default in previous releases was 1. In IMS V10 the default depends on the MINVERS value. MINVERS('10.1') sets the default to 6. MINVERS values less than '10.1' sets the default to 1.

The output from a DBRC LIST command includes full precision timestamps. In previous releases, the timestamp in the listing included only tenths of seconds. If you request the time zone offset to be listed, it follows the timestamp. Since the timestamp is larger by five bytes, the offset is moved by five bytes. In general there were at least five blanks following timestamps in listings from previous releases. This means that other information did not have to be moved for full precision timestamps in V10. The exception is the timeline information in the listing created by a LIST.HISTORY command. The timeline at the end of LIST.HISTORY outputs still has only tenths of seconds. There was not room to add the additional five bytes to these timestamps in the listing.


DBRC Timestamp Formats

- External timestamp format in V8 and V9
  - Example: '2007.318 10:30:23.4 PST'
- External timestamp format in V10
  - Example: '2007.318 10:30:23.456789 PST'
    - Omitted trailing digits are assumed to be zeros
- Internal timestamp format is not changed in V10
  - Internal format in V8, V9, and V10 RECONs:
    - yyyydddFhhmmssthijmuFqqs
      - F - hexadecimal 'F'
      - hijmu - hundredths, thousandths, etc. of second
        - hijmu are ignored in V8 and V9
        - hijmu are ignored in V10 unless MINVERS('10.1')
      - qq - quarter hour offset for time zone
      - s - sign for offset

5

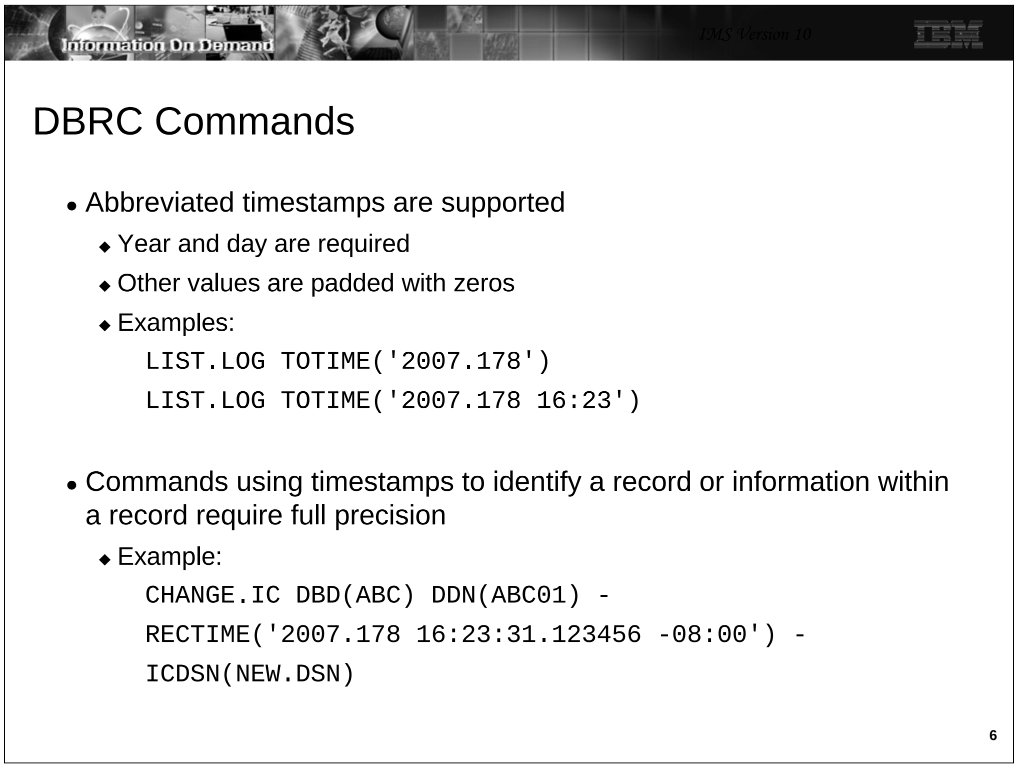
The timestamp used in DBRC commands and utility control statements is expanded in V10. In previous releases it may include times to one tenth of a second. In V10 it may include times to a microsecond.

Compressed and punctuated timestamp formats continue to be used. The change is the additional significant digits for hundredths through millionths of a second. The examples shown here are punctuated timestamps. The corresponding compressed timestamps would be:

V8 or V9:	063181030234 -8
V10:	06318103023456789 -8

The internal format of the timestamp in the RECONs is not changed in V10. The format has included microseconds since IMS V6. The PRILOG start times have had zeros in these positions. Most other RECON records have had actual values. Nevertheless, they have been ignored by other processing. IMS V10 stores actual values in all time fields. When MINVERS('10.1') is used with IMS V10, these positions become significant and are not ignored.

The internal timestamp format is used by the DBRC API. It has not changed in V10.



DBRC Commands

- Abbreviated timestamps are supported
  - Year and day are required
  - Other values are padded with zeros
  - Examples:

```
LIST.LOG TOTIME('2007.178')
LIST.LOG TOTIME('2007.178 16:23')
```

- Commands using timestamps to identify a record or information within a record require full precision
  - Example:

```
CHANGE.IC DBD(ABC) DDN(ABC01) -
RECTIME('2007.178 16:23:31.123456 -08:00') -
ICDSN(NEW.DSN)
```

6

The first example lists all log records or OLDS entries that have start times on or before day 178 in 2007. Full precision is not required on the TOTIME value. The next example lists all log records or OLDS entries that have start time on or before day 16:28 (4:28 p.m.) on day 178 in 2007.

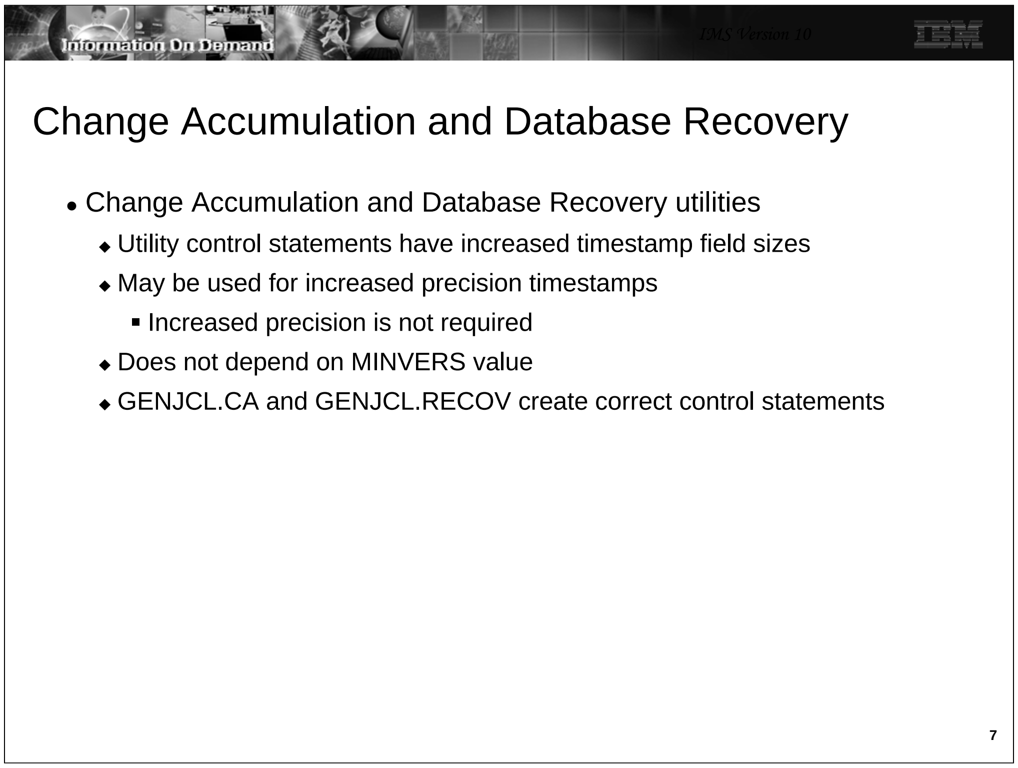
Many RECON records include the timestamp as their keys. When a RECON record is recorded with a full precision timestamp, commands for that specific record require full precision in their timestamps. The second example shows a command where the full precision timestamp is required. An image copy record was created with a full precision timestamp. The timestamp is part of the key. The CHANGE.IC command in the example changes the data set name of the image copy of the database data set with DDNAME ABC01 in the ABC database which was run at the time specified in the RECTIME parameter.

Commands requiring full precision timestamps are

• CHANGE.BKOUT, CHANGE.CA, CHANGE.IC, CHANGE.PRILOG RLDS, CHANGE.PRILOG SLDS, CHANGE.PRILOG TSLDS, CHANGE.SECLOG RLDS, CHANGE.SECLOG SLDS, CHANGE.SECLOG TSLDS, and CHANGE.UIC

• DELETE.CA, DELETE.IC, DELETE.RECOV, DELETE.REORG, and DELETE.UIC

If you enter a command with a timestamp that has with less than six digits for microseconds, DBRC uses zeros for the missing digits. Since NOTIFY commands create new information in the RECONs, their timestamps do not have to match existing data. You can use timestamps without all six digits for microseconds for NOTIFY commands. Zeros will be used for any of the missing digits.



Information On Demand

IMS Version 10

Change Accumulation and Database Recovery

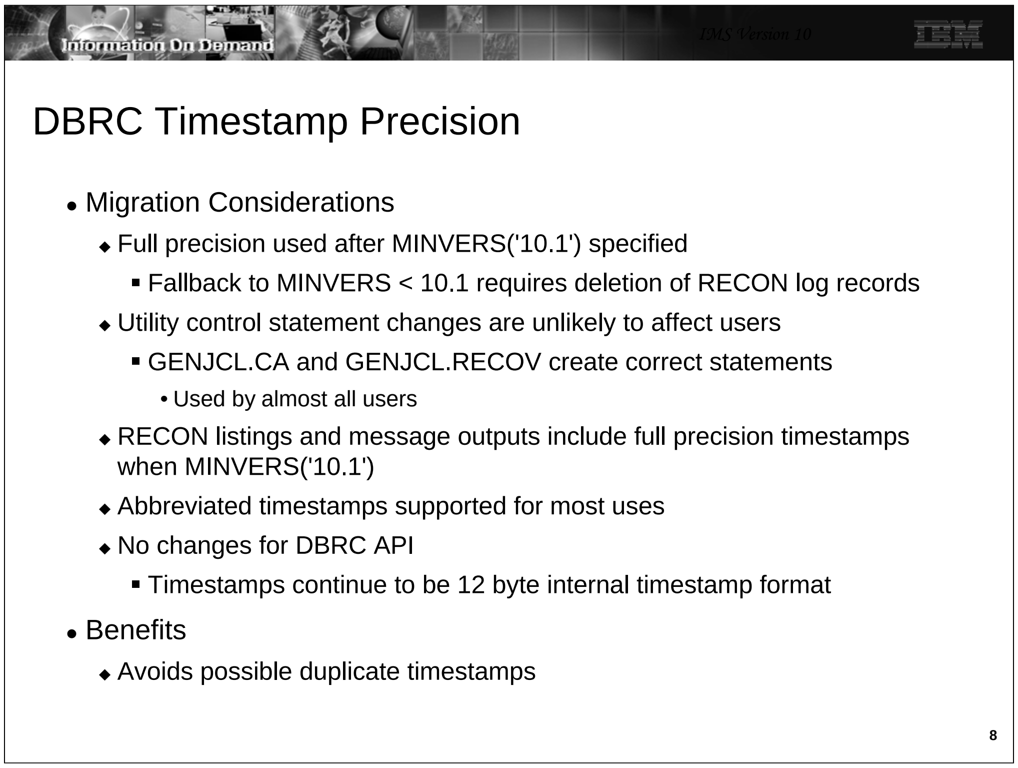
- Change Accumulation and Database Recovery utilities
  - Utility control statements have increased timestamp field sizes
  - May be used for increased precision timestamps
    - Increased precision is not required
  - Does not depend on MINVERS value
  - GENJCL.CA and GENJCL.RECOV create correct control statements

The Database Change Accumulation utility DB0 and DB1 control statements have been modified to support timestamps with greater precision. The new expanded timestamp format may be used in V10 but it is not required. DB0 control statements are used to specify the database data sets that are accumulated to the new change accumulation data set. DB1 control statements are used to specify the database data sets that are written to the new output log data set. Since the timestamps now may occupy more columns in the control statements, the position of the database data set DDNAMEs have moved and only three may be specified on one control statement. Previous IMS versions allowed four to be specified on a control statement.

The S control statement for the Database Recovery utility is used to specify the database and DDNAME for the database data set that is to be recovered. If the recovery is a timestamp recovery, the timestamp is also specified. The new expanded timestamp format may be used in V10 but it is not required. In previous releases column 57 was used for an indicator. The indicator could specify that a user image copy had been restored or that an RSR receive was done. Since the timestamp may use column 57, the indicator is now coded in column 63 when needed. There is another small change in the coding of the timestamp. In previous releases there was not a space between the time and the sign for the time zone offset.

GENJCL.CA has been updated to create the new format of the control statements. These changes to the control statements will have no effects on users who create Change Accumulation JCL and control statements with GENJCL.CA. This is the vast majority of IMS installations.

GENJCL.RECOV has been updated to create the new format of the control statement for timestamp recovery. The change in the control statement will have no effects on users who create Database Recovery JCL and control statements with GENJCL.RECOV. This is the vast majority of IMS installations.


Information On Demand
IMS Version 10
IBM

DBRC Timestamp Precision

- Migration Considerations
  - Full precision used after MINVERS('10.1') specified
    - Fallback to MINVERS < 10.1 requires deletion of RECON log records
  - Utility control statement changes are unlikely to affect users
    - GENJCL.CA and GENJCL.RECOV create correct statements
      - Used by almost all users
  - RECON listings and message outputs include full precision timestamps when MINVERS('10.1')
  - Abbreviated timestamps supported for most uses
  - No changes for DBRC API
    - Timestamps continue to be 12 byte internal timestamp format
- Benefits
  - Avoids possible duplicate timestamps

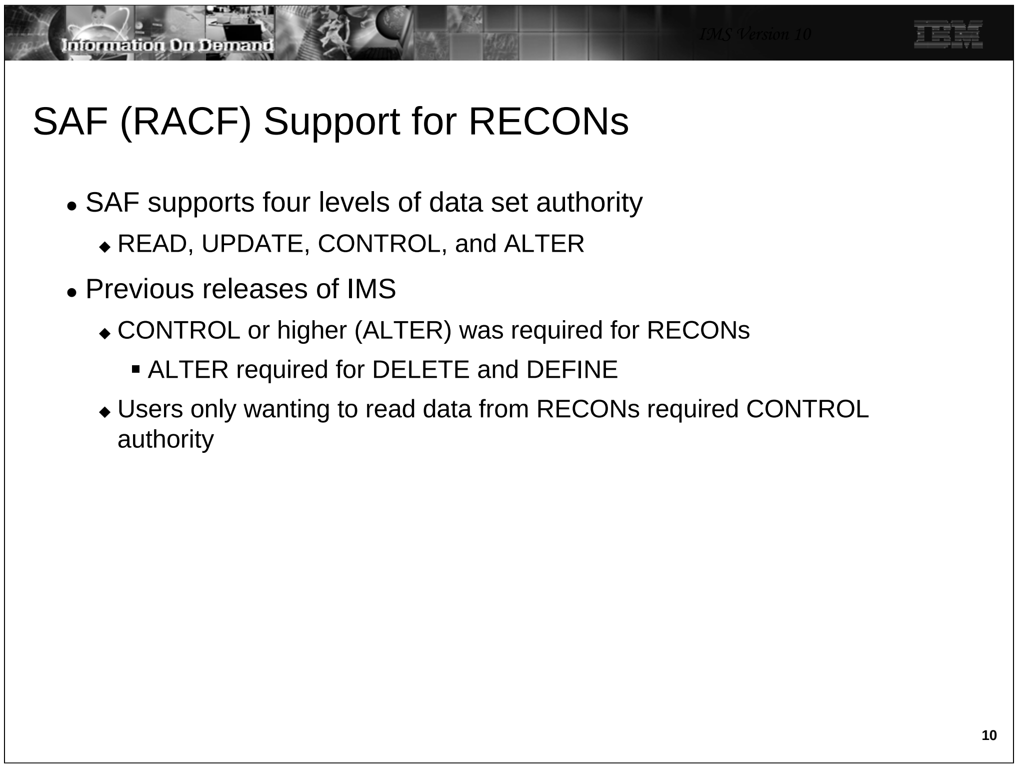
8

Full precision timestamps are not implemented unless the RECONs have MINVERS('10.1') specified. Even when MINVERS is set to a lower value, the IMS V10 Change Accumulation and Database Recovery utility control statements require new formats which accommodate full precision timestamps. Nevertheless, this is unlikely to be a concern to users since GENJCL.CA and GENJCL.RECOV in IMS V10 always produce control statements with the V10 formats.

If you wish, you may specify abbreviated timestamps for most uses. DBRC will interpret the time correctly. Full precision timestamps are required in CHANGE and DELETE commands when a full precision timestamp is part of the RECON record key.

When MINVERS('10.1') is set log records are created with the full precision timestamps. If you fallback to MINVERS('9.1') or MINVERS('8.1'), the log records created with full precision timestamps must be deleted before the CHANGE.RECON MINVERS(…) command is issued. The records that must be deleted include PRILOG, PRISLD, PRIOLD, etc. and the ALLOC records associated with these log records.

RECON READONLY Access and SAF Enhancements



Information On Demand

IMS Version 10

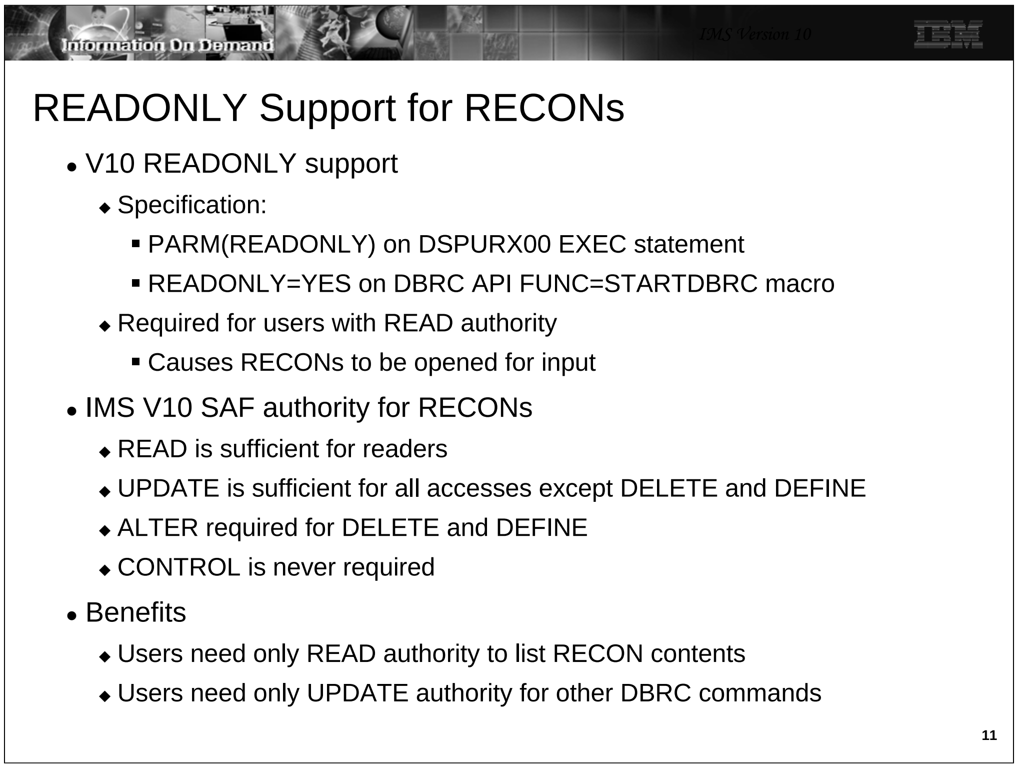
SAF (RACF) Support for RECONs

- SAF supports four levels of data set authority
  - READ, UPDATE, CONTROL, and ALTER
- Previous releases of IMS
  - CONTROL or higher (ALTER) was required for RECONs
    - ALTER required for DELETE and DEFINE
  - Users only wanting to read data from RECONs required CONTROL authority

10

System Authorization Facility (SAF) products, such as RACF, support four levels of data set authority. In ascending sequence of authority these are READ, UPDATE, CONTROL, and ALTER. Previous releases of IMS required at least CONTROL authority for all users of the RECONs. IDCAMS DEFINE and DELETE of a RECON data set required ALTER.

Previous IMS releases opened the RECONs for update with CONTROL specified in the VSAM ACB for the RECONs. This required CONTROL authority for open. This was true even if the user only wanted to read the RECONs as would be done for a LIST command.



READONLY Support for RECONs

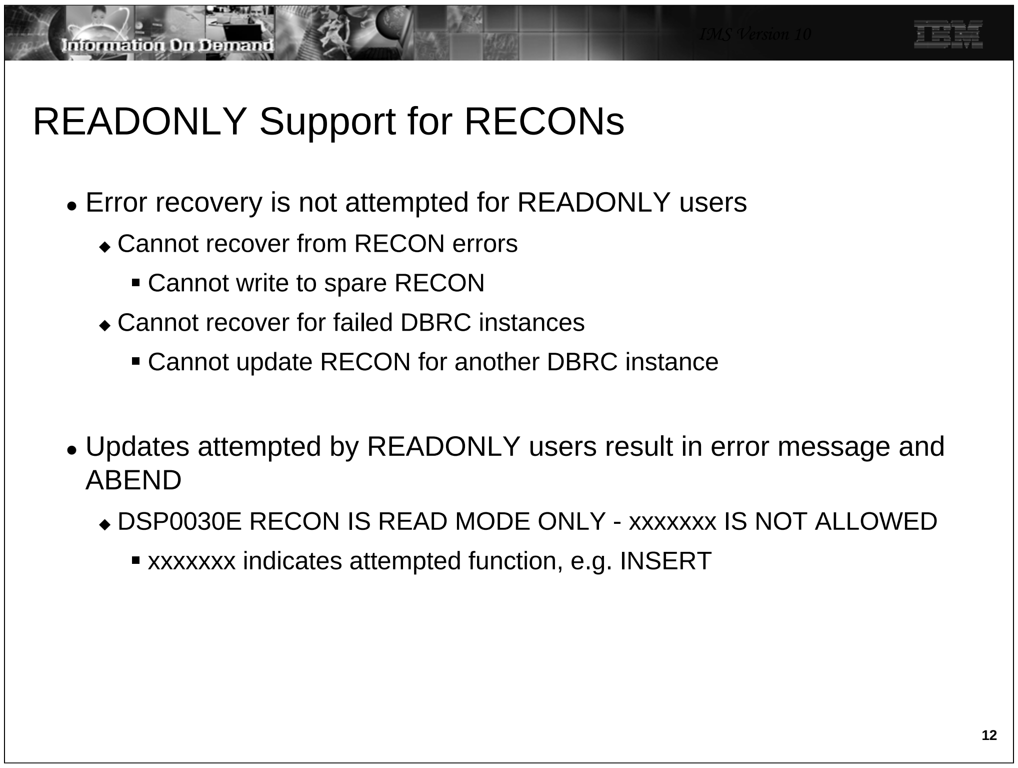
- V10 READONLY support
  - Specification:
    - PARM(READONLY) on DSPURX00 EXEC statement
    - READONLY=YES on DBRC API FUNC=STARTDBRC macro
  - Required for users with READ authority
    - Causes RECONs to be opened for input
- IMS V10 SAF authority for RECONs
  - READ is sufficient for readers
  - UPDATE is sufficient for all accesses except DELETE and DEFINE
  - ALTER required for DELETE and DEFINE
  - CONTROL is never required
- Benefits
  - Users need only READ authority to list RECON contents
  - Users need only UPDATE authority for other DBRC commands

11

IMS V10 adds READONLY support for the RECONs. This is invoked with PARM(READONLY) on the EXEC statement for the DBRC utility (DSPURX00) or by specifying the new READONLY=YES parameter on the DBRC API FUNC=STARTDBRC macro. When READONLY is specified, the RECONs are opened for read. This means that only READ authority is required in SAF (RACF).

IMS V10 has made another change to open. Due to this change, CONTROL does not have to be specified for users who update the RECONs. Only UPDATE authority is required. Of course, ALTER is still required for users who DELETE and DEFINE the data sets. In previous releases DBRC opened the RECONs with CONTROL specified in the VSAM ACB for the RECONs. This required CONTROL authority for open. In IMS V10 the open has changed. If READONLY is not specified, the open is done for update but CONTROL is not specified in the ACB. This means that only UPDATE authority is required.

As with previous releases, if you invoke the DBRC utility from your program you may use the DSPURXRT entry point. IMS V10 has added the capability to specify READONLY through a parameter passed to the entry point in the first word of the argument list.



READONLY Support for RECONs

- Error recovery is not attempted for READONLY users
  - Cannot recover from RECON errors
    - Cannot write to spare RECON
  - Cannot recover for failed DBRC instances
    - Cannot update RECON for another DBRC instance

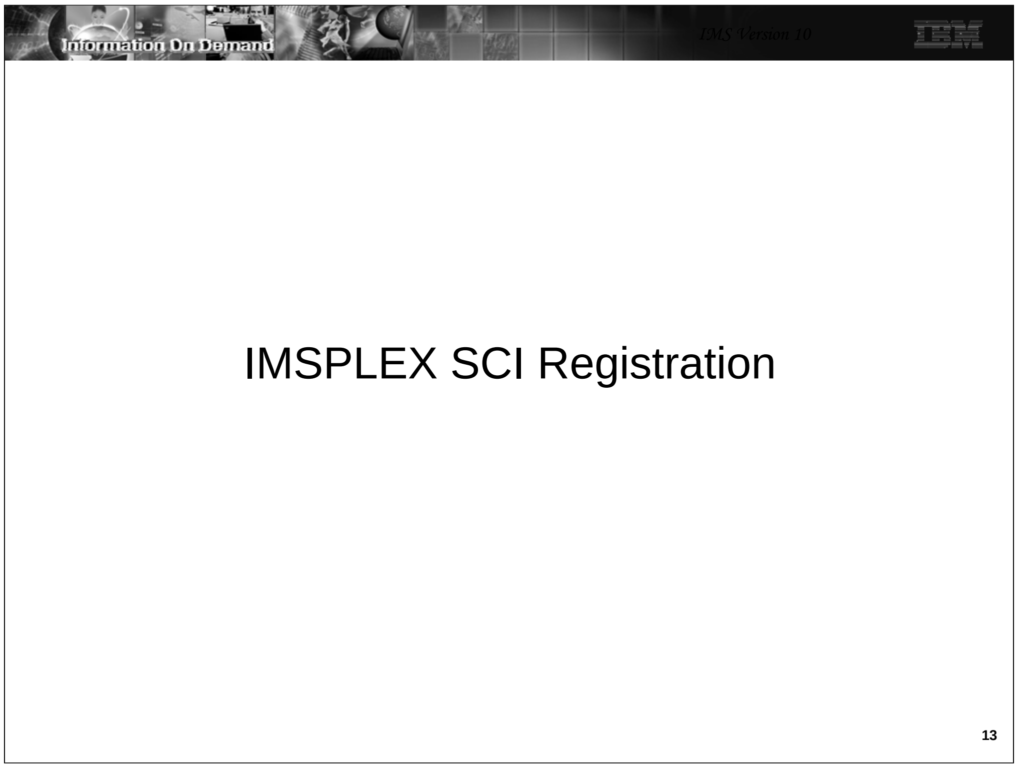
- Updates attempted by READONLY users result in error message and ABEND
  - DSP0030E RECON IS READ MODE ONLY - xxxxxxx IS NOT ALLOWED
    - xxxxxxx indicates attempted function, e.g. INSERT

12

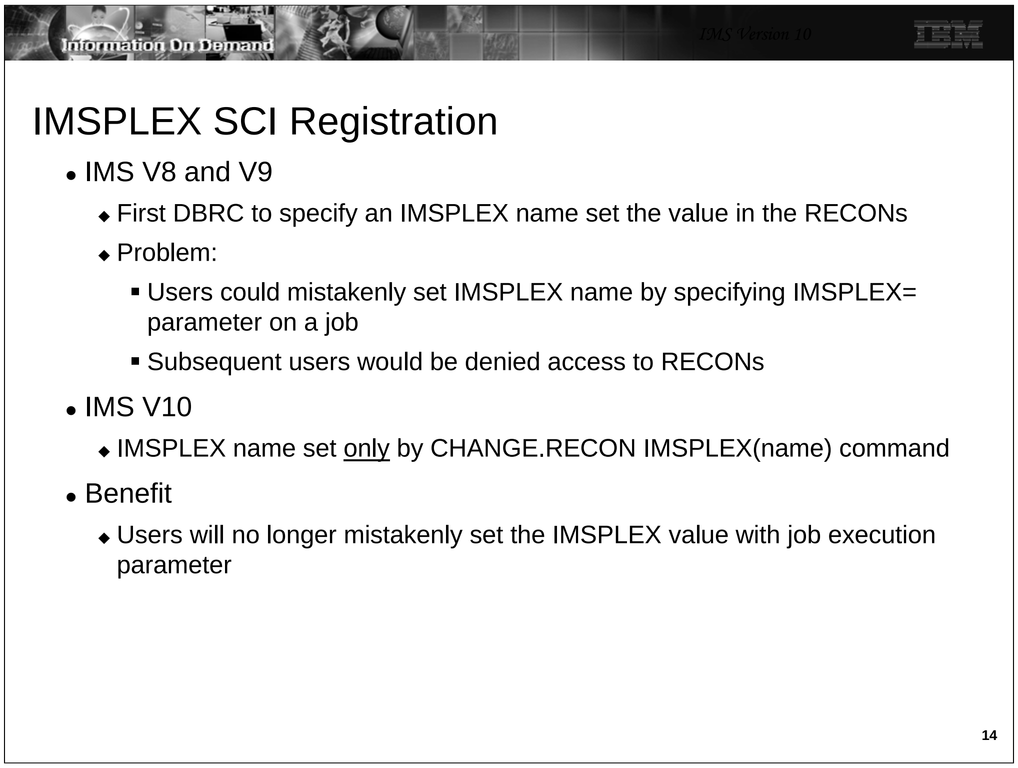
Since READONLY causes IMS to open the RECONs for input, users of READONLY cannot invoke recovery processes for the RECONs. There are two kinds of recovery processes. The first is recovery from RECON errors. If an I/O error occurs on a RECON data set, updaters can reconfigure the RECONs. This includes copying the good RECON to the spare. READONLY users cannot do writes, so they cannot do this recovery process. The second recovery process is recovering from a failed DBRC instance. When a DBRC instance (batch job, utility, or online system) updates multiple RECON records it first writes a Multiple Update Record (MUP) to the RECONs. It then does the updates and, finally, deletes the MUP record. If it fails in the middle of this process, another DBRC instance recovers. The other DBRC reads the MUP record and either completes or backs out the changes. If the other DBRC instance is a READONLY user, it cannot perform this recovery because it cannot write.

If a READONLY execution attempts to update the RECONs, message DSP0030E is issued and the application abends. The variable text in the message indicates the operation that was attempted. The possible values are:

CONFIG	An attempt was made to reconfigure the RECON data sets.
DELETE	An attempt was made to delete a record from the RECON.
INSERT	An attempt was made to create a new RECON record.
UPDATE	An attempt was made to update an existing RECON record.
UPGRADE	An attempt was made to upgrade the RECON data sets.
MODE-SW	An attempt was made to switch accessing mode.



IMSPLEX SCI Registration



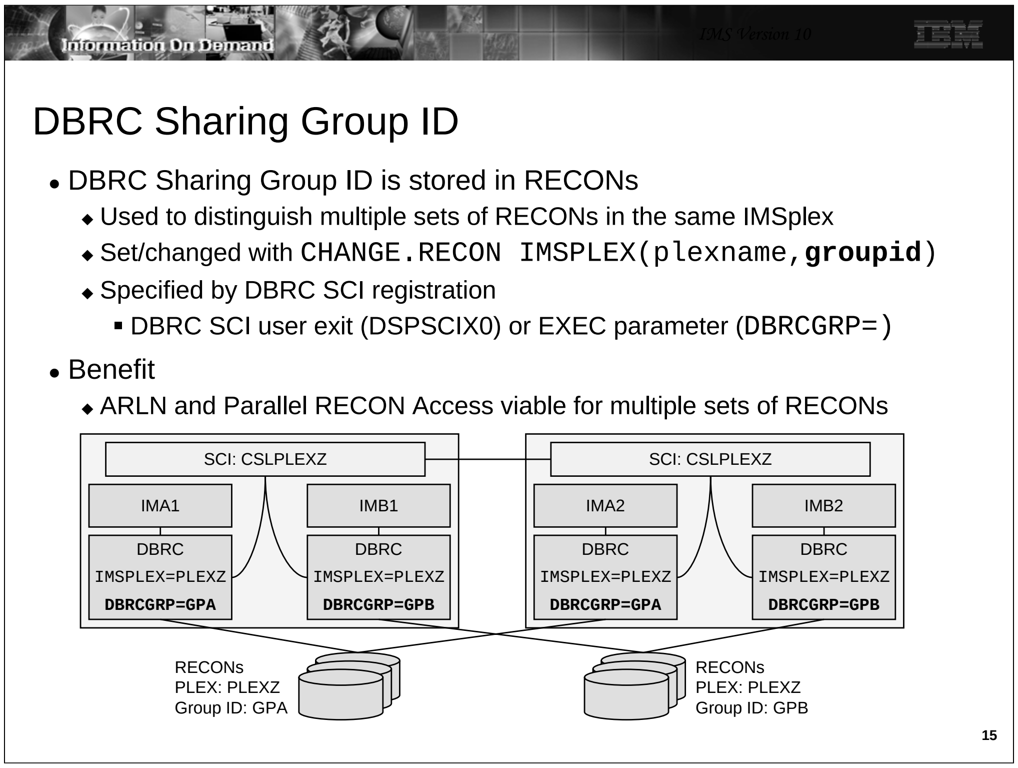
IMSPLEX SCI Registration

- IMS V8 and V9
  - First DBRC to specify an IMSPLEX name set the value in the RECONs
  - Problem:
    - Users could mistakenly set IMSPLEX name by specifying IMSPLEX= parameter on a job
    - Subsequent users would be denied access to RECONs
- IMS V10
  - IMSPLEX name set only by CHANGE.RECON IMSPLEX(name) command
- Benefit
  - Users will no longer mistakenly set the IMSPLEX value with job execution parameter

14

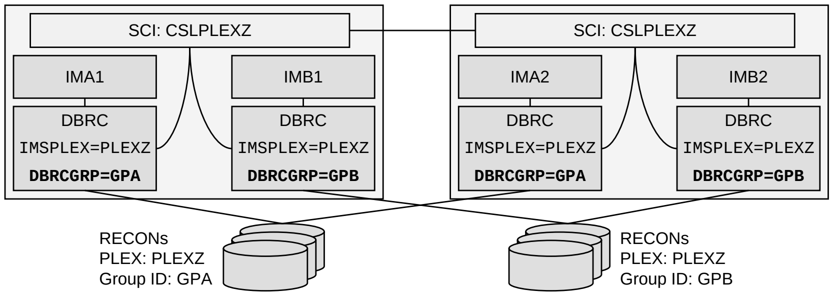
When the IMSPLEX name is set in the RECONs, all subsequent users must specify the same IMSPLEX name. They must either include the IMSPLEX= execution parameter with the correct value or the DBRC SCI Registration exit must specify the correct value.

In previous releases some installations reported that users mistakenly specified the IMSPLEX= execution parameter. This caused subsequent jobs to fail until the IMSPLEX name could be removed from the RECONs with a CHANGE.RECON NOPLEX command. The change in IMS V10 will prevent this situation from occurring. Now the IMSPLEX name will only be set by the CHANGE.RECON IMSPLEX(name) command.


Information On Demand
IMS Version 10

DBRC Sharing Group ID

- DBRC Sharing Group ID is stored in RECONs
  - Used to distinguish multiple sets of RECONs in the same IMSplex
  - Set/changed with `CHANGE.RECON IMSPLEX(plexname,`**`groupid`**`)`
  - Specified by DBRC SCI registration
    - DBRC SCI user exit (DSPSCIX0) or EXEC parameter (`DBRCGRP=`)
- Benefit
  - ARLN and Parallel RECON Access viable for multiple sets of RECONs



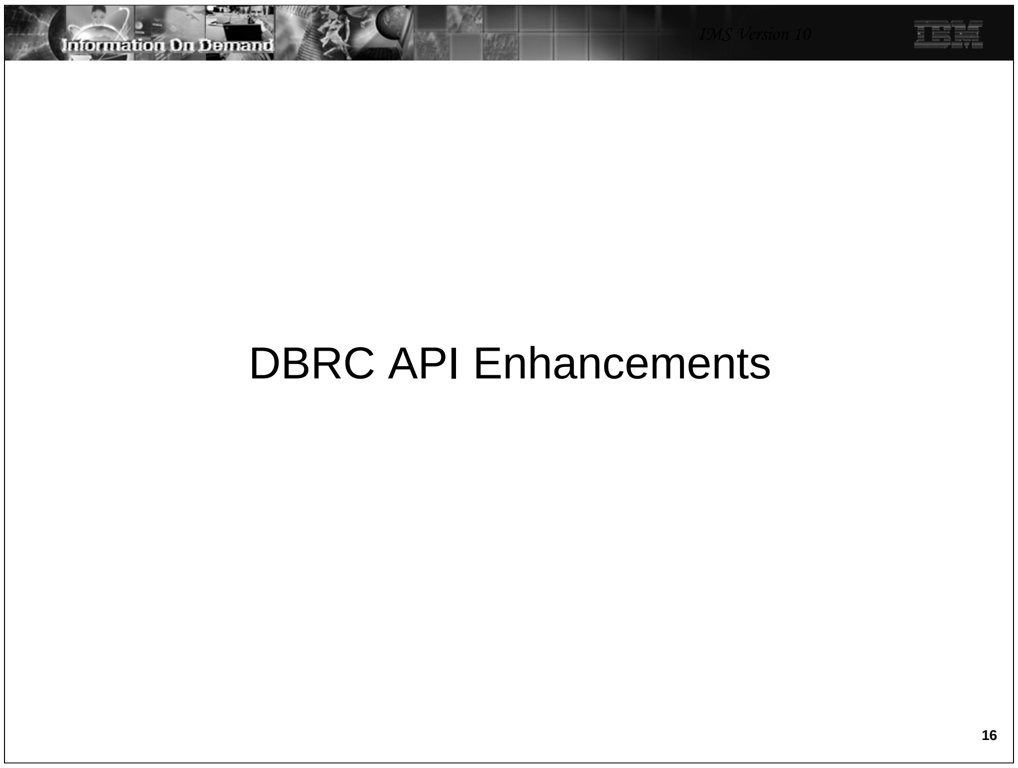
The DBRC Group ID is new in IMS V10. It is used so that Automatic RECON Loss Notification (ARLN) can distinguish between the different sets of RECONs. The IDs ensure that a reconfiguration of the RECONs is only processed by the DBRCs using that set of RECONs.

When upgrading RECONs from a previous release where the plexname was specified, the group ID defaults to 001. If there is no plexname in the RECONs when they are upgraded, the DBRC Group ID is not set.

The DBRC Group ID is set or changed with CHANGE.RECON IMSPLEX(plexname,groupid) command. If the DBRC Group ID is not set before the command is issued, it defaults to 001.

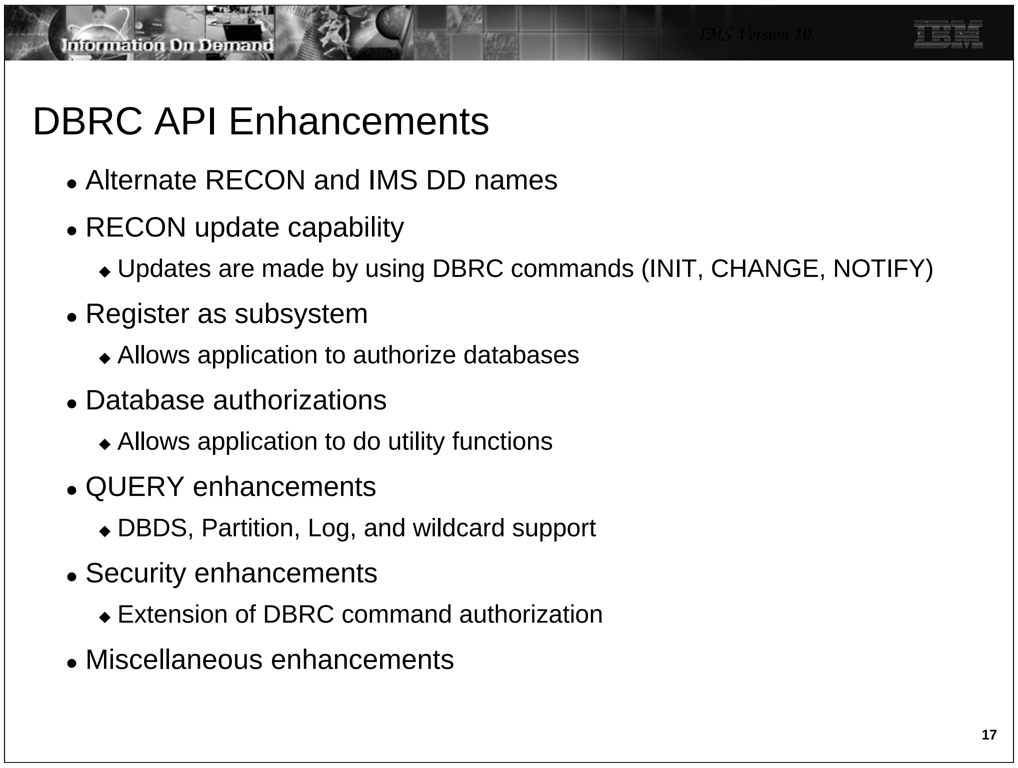
DBRC SCI registration must include the DBRC Group ID along with the IMSplex name.

The example on this slide shows an IMSPLEX with two sets of RECONs. One set uses the DBRC Group ID of GPA. The other set uses GPB. The DBRCs using the GPA RECONs must specify GPA to SCI registration. The DBRCs using the GPB RECONs must specify GPB to SCI registration.



DBRC API Enhancements

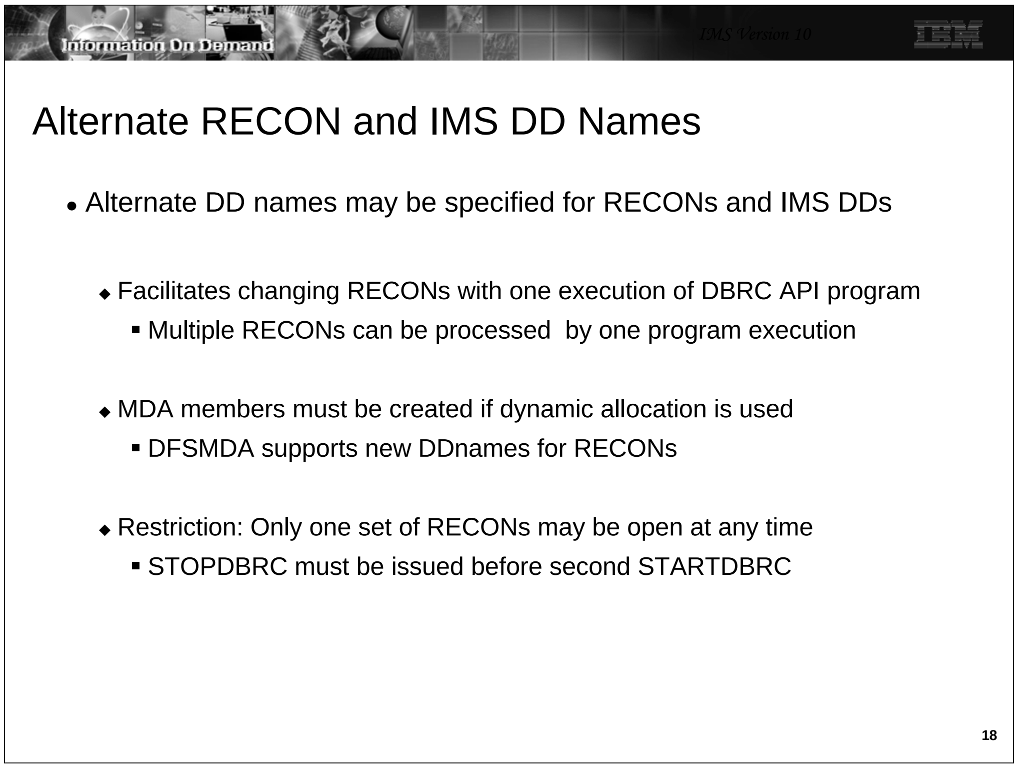
IMS V10 introduces several enhancements to the DBRC API. This class provides an overview of these enhancements. You should refer to the IMS V10 System Programming API Reference manual for specific information on using this API.



DBRC API Enhancements

- Alternate RECON and IMS DD names
- RECON update capability
  - Updates are made by using DBRC commands (INIT, CHANGE, NOTIFY)
- Register as subsystem
  - Allows application to authorize databases
- Database authorizations
  - Allows application to do utility functions
- QUERY enhancements
  - DBDS, Partition, Log, and wildcard support
- Security enhancements
  - Extension of DBRC command authorization
- Miscellaneous enhancements

The DBRC API is described in the IMS V10 *System Programming API Reference* manual. It contains details on the DSPAPI macro, its parameters and usage, and on the control blocks created by DBRC API requests.



Alternate RECON and IMS DD Names

- Alternate DD names may be specified for RECONs and IMS DDs
  - Facilitates changing RECONs with one execution of DBRC API program
    - Multiple RECONs can be processed by one program execution
  - MDA members must be created if dynamic allocation is used
    - DFSMDA supports new DDnames for RECONs
  - Restriction: Only one set of RECONs may be open at any time
    - STOPDBRC must be issued before second STARTDBRC

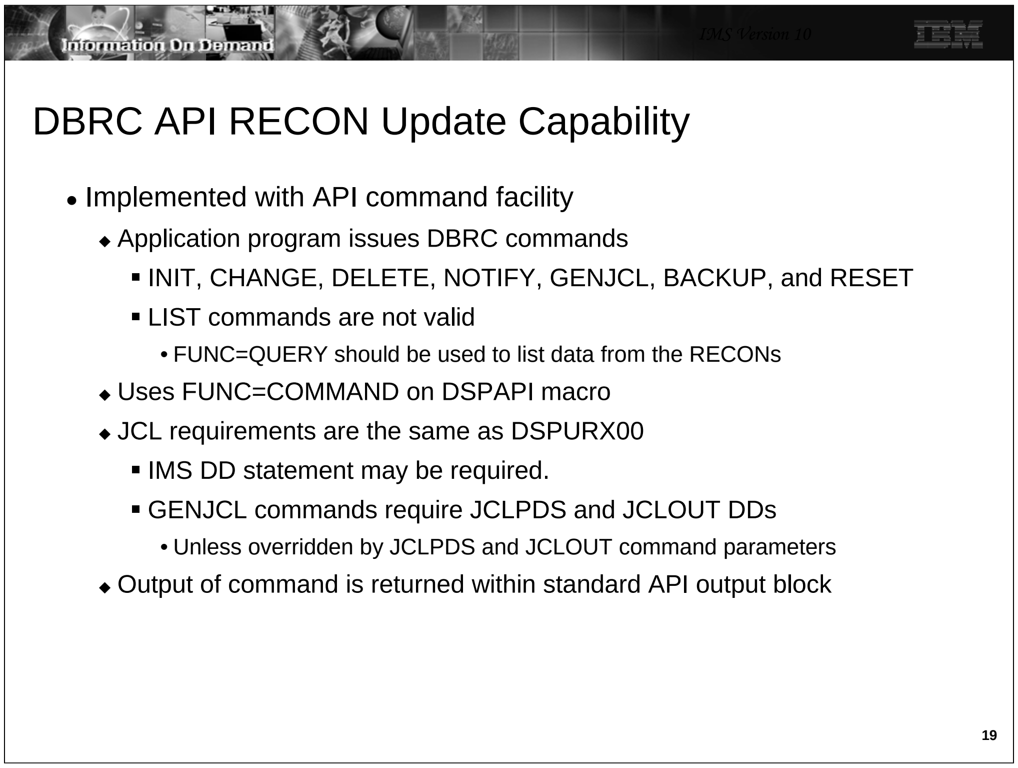
18

Programs using the DBRC API may use alternate names for the IMS (DBDLIB) and RECON DDNAMEs. This could make it easier to develop programs which access multiple sets of RECONs. In V9 these programs had to use dynamic allocation (SVC 99) to change the data sets for RECON1, RECON2, and RECON3 to access a different set of RECONs. In IMS V10 they may use JCL to allocate multiple sets or they may use DFSMDA to create dynamic allocation members for the different RECONs.

The DFSMDA macro has been enhanced to support the alternate DD names for RECONs. A new parameter, DDNAME=, has been added to the DFSMDA macro for TYPE=RECON.

Programs using the DBRC API are still restricted to accessing only one set of RECONs at a time. A FUNC=STOPDBRC request must be processed before a second set of RECONs may be opened with a second FUNC=STARTDBRC request.

As with previous releases, if you invoke the DBRC utility from your program you may use the DSPURXRT entry point. IMS V10 has added the capability to specify alternate DD names for the RECON data sets. This is done through the expansion of the list of DDNAMES passed to the entry point in the second word of the argument list.



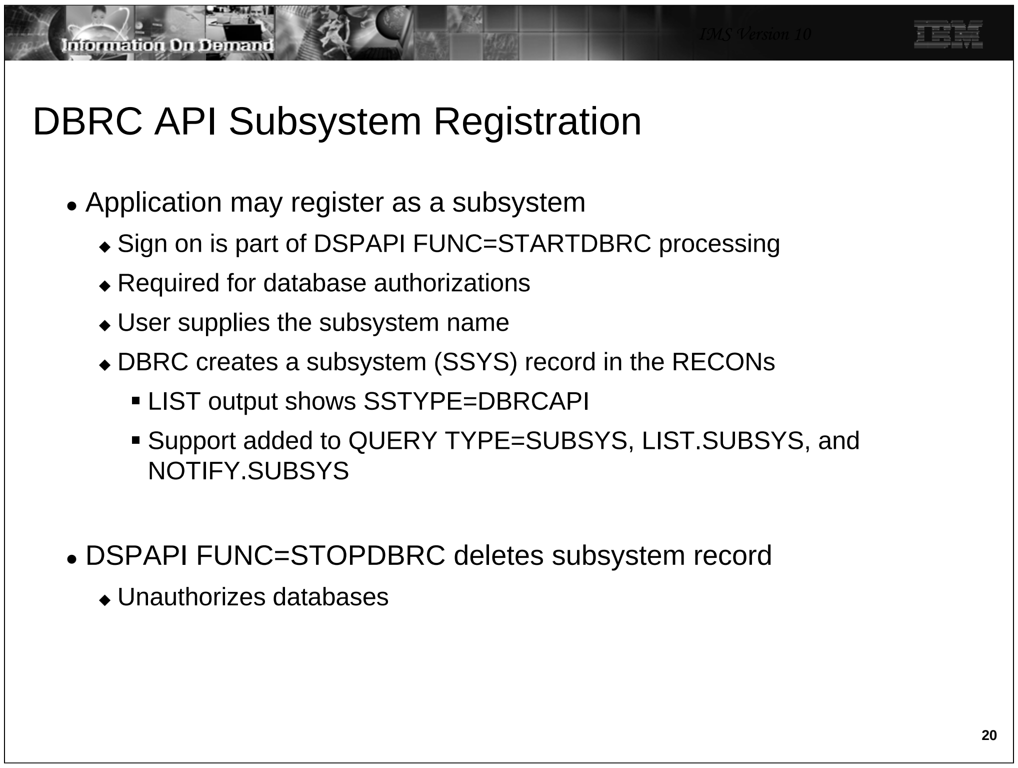
DBRC API RECON Update Capability

- Implemented with API command facility
  - Application program issues DBRC commands
    - INIT, CHANGE, DELETE, NOTIFY, GENJCL, BACKUP, and RESET
    - LIST commands are not valid
      - FUNC=QUERY should be used to list data from the RECONs
  - Uses FUNC=COMMAND on DSPAPI macro
  - JCL requirements are the same as DSPURX00
    - IMS DD statement may be required.
    - GENJCL commands require JCLPDS and JCLOUT DDs
      - Unless overridden by JCLPDS and JCLOUT command parameters
  - Output of command is returned within standard API output block

19

The DBRC API has been extended. It was introduced in IMS V9. The original implementation of the DBRC API allowed you to query the data in the RECONs by issuing DSPAPI macros. The extensions to the API in V10 allow you to update the RECONs. Updates are done by allowing invoking DBRC commands. Commands such as INIT, CHANGE, DELETE, NOTIFY, and RESET may be used to update the RECONs. This interface may also be used to issue GENJCL and BACKUP commands which do not update the RECONs. LIST commands are not valid. The DSPAPI FUNC=QUERY function should be used to list data from the RECONs.

The output from commands is returned in an API output block. This block has the same format as the QUERY blocks that were introduced in V9. Details of the command blocks are shown later.



DBRC API Subsystem Registration

- Application may register as a subsystem
  - Sign on is part of DSPAPI FUNC=STARTDBRC processing
  - Required for database authorizations
  - User supplies the subsystem name
  - DBRC creates a subsystem (SSYS) record in the RECONs
    - LIST output shows SSTYPE=DBRCAPI
    - Support added to QUERY TYPE=SUBSYS, LIST.SUBSYS, and NOTIFY.SUBSYS
- DSPAPI FUNC=STOPDBRC deletes subsystem record
  - Unauthorizes databases

20

The STARTDBRC function is enhanced to support the creation of subsystem records. The user application may register with DBRC as a subsystem. This is required for authorizing databases. Database authorization support has also been added to the DBRC API. It will be explained later. Subsystem registration is done by adding an SSID= parameter to the DSPAPI FUNC=STARTDBRC macro. The meanings of the other parameters on the macro are unchanged.

A new type of subsystem has been added to the those that are stored in the SUBSYS record. This type is DBRCAPI. It is used when DSPAPI FUNC=STARTDBRC is used to create the SUBSYS record.

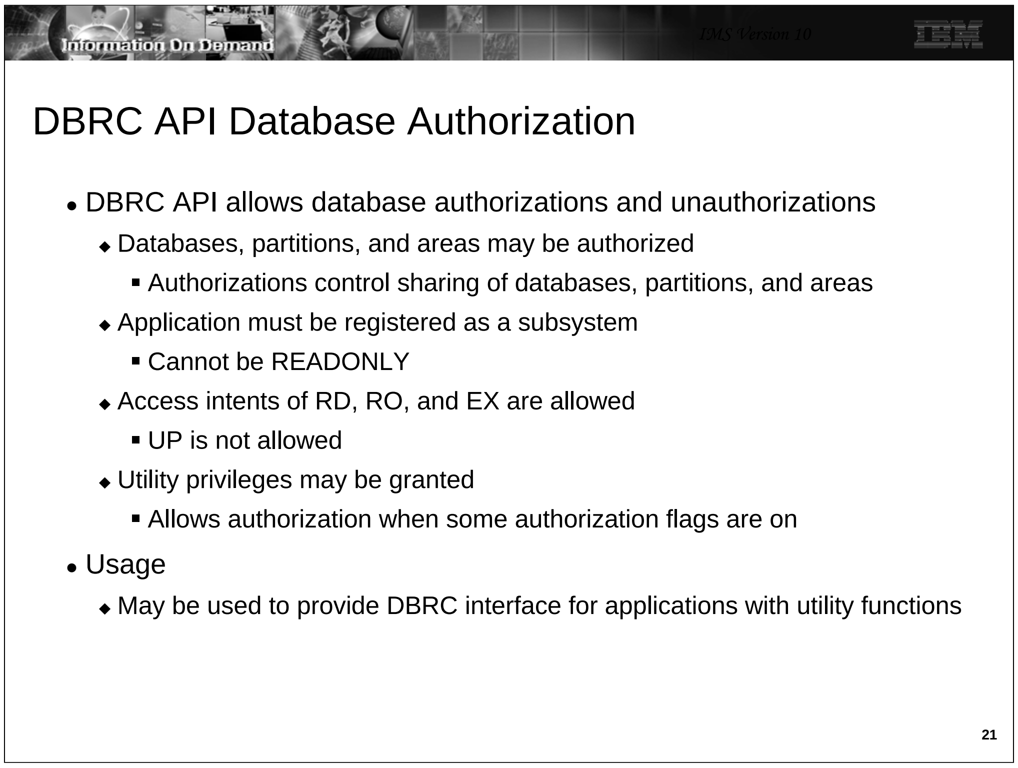
Support for this new type has been added to QUERY TYPE=SUBSYS, LIST.SUBSYS, and NOTIFY.SUBSYS.

QUERY TYPE=SUBSYS: This QUERY request has been enhanced to include SSTYPE=DBRCAPI to limit the returned information to only DBRC API subsystems. A new flag bit meaning has been added to the output block created by a QUERY TYPE=SUBSYS. It indicates that the subsystem is a DBRC API subsystem. The bit is only set when VERSION=2.0 is specified on the QUERY.

LIST.SUBSYS and LIST.RECON: The output of the LIST.SUBSYS and LIST.RECON commands includes SSTYPE=DBRCAPI for DBRC API subsystems. You can list only DBRCAPI subsystems by using the DBRCAPI keyword on the LIST.SUBSYS command.

NOTIFY.SUBSYS: The NOTIFY.SUBSYS command has been enhanced. The new DBRCAPI keyword causes the command to create a DBRC API subsystem record.

The DBRCAPI subsystem records are deleted by DSPAPI FUNC=STOPDBRC macros. If any databases are authorized to the subsystem, they are unauthorized when the macro is processed.



DBRC API Database Authorization

- DBRC API allows database authorizations and unauthorizations
  - Databases, partitions, and areas may be authorized
    - Authorizations control sharing of databases, partitions, and areas
  - Application must be registered as a subsystem
    - Cannot be READONLY
  - Access intents of RD, RO, and EX are allowed
    - UP is not allowed
  - Utility privileges may be granted
    - Allows authorization when some authorization flags are on
- Usage
  - May be used to provide DBRC interface for applications with utility functions

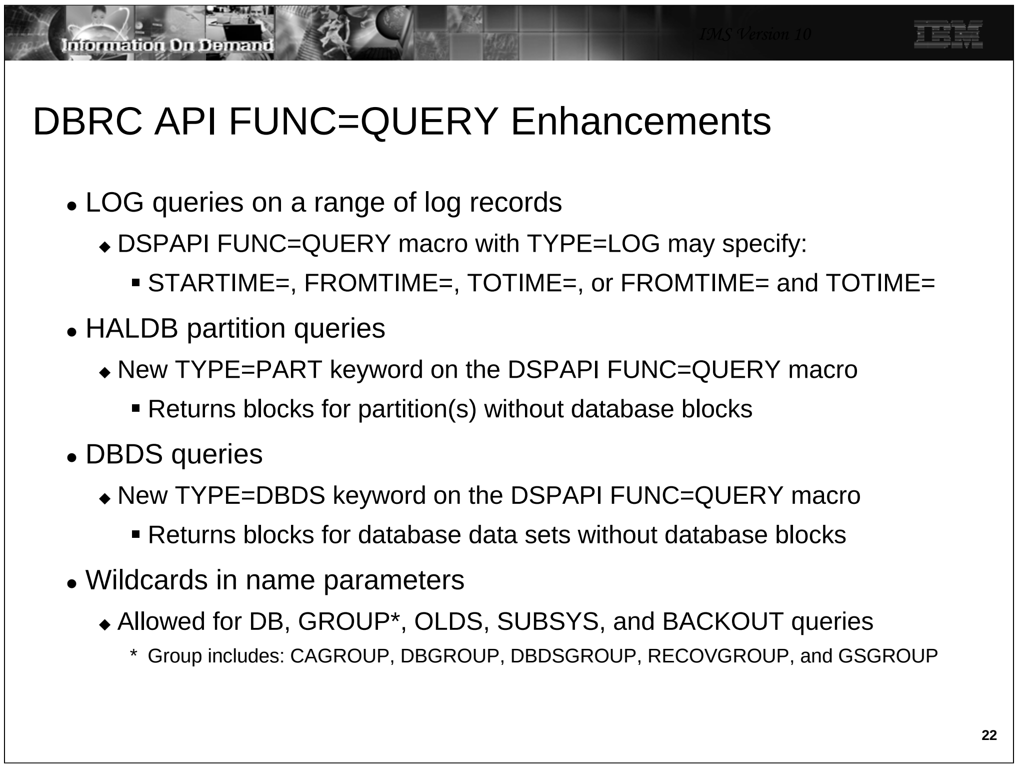
21

Applications using the DBRC API may authorize and unauthorize databases, partitions, and areas. Databases are authorized to subsystems, so authorization requests come from applications that are registered as subsystems. Since authorizations update the RECONs, the application cannot be using READONLY.

When a database, partition, or area is authorized, its access intent may be specified. Block level data sharing is not supported for DBRC API users, therefore, UP is not allowed. If any changes are to be made to a database, partition, or area, EX access must be used.

Utility privileges may be granted with an authorization. This allows authorizations even though the 'Prohibit Further Authorization,' 'Image Copy Needed,' or 'Read Only' flags are on. This is explained further on the next page.

Authorizations may be needed when the DBRC API application is providing functions similar to IMS database utilities, such as back up, recovery, or reorganization.


Information On Demand
IMS Version 10

DBRC API FUNC=QUERY Enhancements

- LOG queries on a range of log records
  - DSPAPI FUNC=QUERY macro with TYPE=LOG may specify:
    - STARTIME=, FROMTIME=, TOTIME=, or FROMTIME= and TOTIME=
- HALDB partition queries
  - New TYPE=PART keyword on the DSPAPI FUNC=QUERY macro
    - Returns blocks for partition(s) without database blocks
- DBDS queries
  - New TYPE=DBDS keyword on the DSPAPI FUNC=QUERY macro
    - Returns blocks for database data sets without database blocks
- Wildcards in name parameters
  - Allowed for DB, GROUP*, OLDS, SUBSYS, and BACKOUT queries

    * Group includes: CAGROUP, DBGROUP, DBDSGROUP, RECOVGROUP, and GSGROUP

22

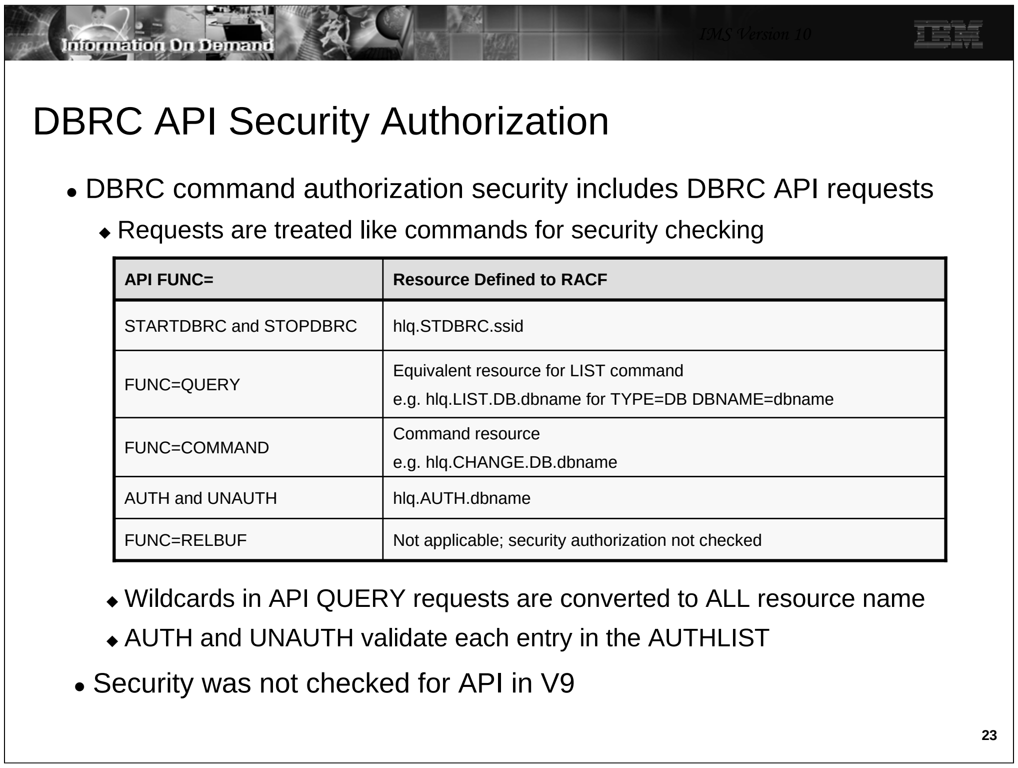
The DBRC API QUERY capability has been enhanced in several ways.

Queries for logs may request log records from a range of times. Queries for HALDB information may request data for partitions without requesting database data.

Database data set data may requested without requesting the data for the database in which the data set or data sets reside.

A wildcard capability has been added to several query types. This is the use of an asterisk (*) when specifying a name. The queries where this may be used are:

Query TYPE=	**Asterisk allowed with**
DB	DBNAME=
*GROUP	NAME=, GROUP=
OLDS	SSID=
SUBSYS	SSID=
BACKOUT	SSID=


DBRC API Security Authorization

- DBRC command authorization security includes DBRC API requests
  - Requests are treated like commands for security checking

API FUNC=	Resource Defined to RACF
STARTDBRC and STOPDBRC	hlq.STDBRC.ssid
FUNC=QUERY	Equivalent resource for LIST command e.g. hlq.LIST.DB.dbname for TYPE=DB DBNAME=dbname
FUNC=COMMAND	Command resource e.g. hlq.CHANGE.DB.dbname
AUTH and UNAUTH	hlq.AUTH.dbname
FUNC=RELBUF	Not applicable; security authorization not checked

  - Wildcards in API QUERY requests are converted to ALL resource name
  - AUTH and UNAUTH validate each entry in the AUTHLIST
- Security was not checked for API in V9

23

DBRC command security was introduced in IMS V8. It may be used to invoke authorization checking for DBRC commands. RACF, or any SAF security product, may be used. Alternatively, an exit routine may be invoked or both the security product and the exit routine may be invoked. Commands are authorized by defining a resource representing the command. In RACF this is done with an RDEFINE statement.

This authorization is extended to the DBRC API in IMS V10. API requests invoke command authorization checking. Command authorization checking uses resources which are defined to secure specific commands or API requests. Resource names have the following form:

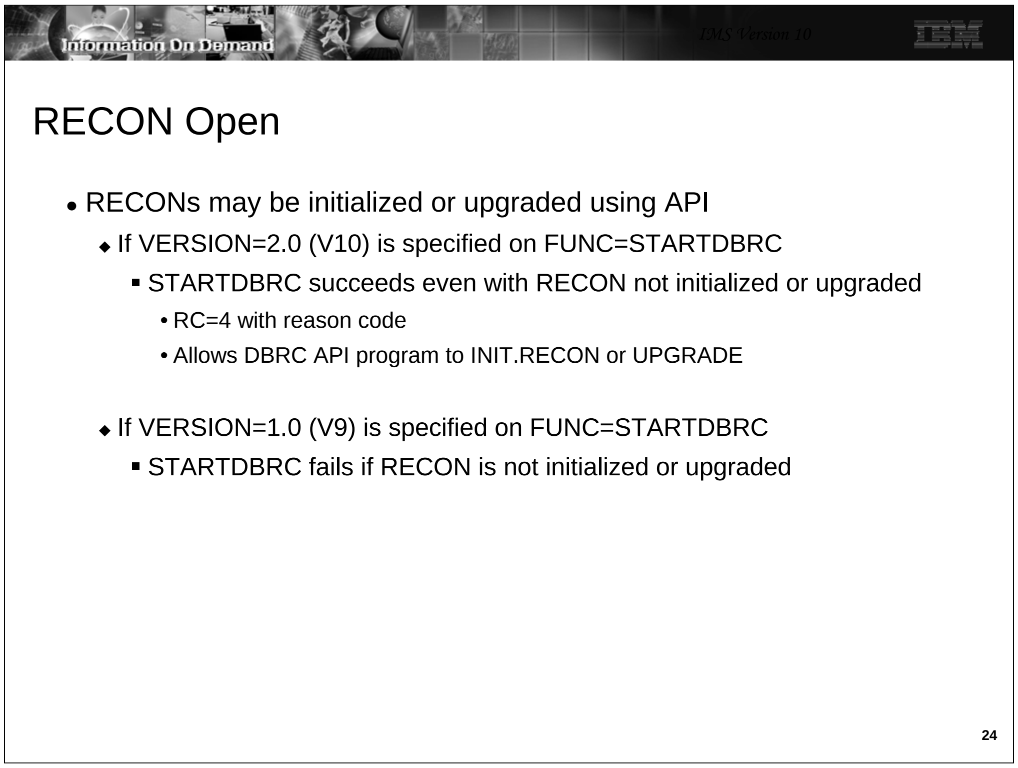
hlq.verb.resoucetype.resourcename

The high level qualifier (hlq) in the resource identifies a set of RECONs. “verb” identifies a DBRC command or API request type. “resourcetype” identifies a resource type on which the command or request operates. For example, a LIST.DB command operates on the database resource type. “resourcetype” is optional. “resourcename” identifies a specific resource instance. For example, a LIST.DB DBD(XYZ) command operates on the XYZ resource instance or name. “resourcename” is optional.

This scheme has been extended for API requests as shown in the table.

TYPE=STARTDBRC and TYPE=STOPDBRC requests are checked using resource of hlq.STDBRC.ssid. ssid specifies a subsystem and is optional. It restricts the use of the requests for a specific subsystem.

Security was not checked for any DBRC API requests in IMS V9. It is possible that a program which executed successfully in IMS V9 will fail security when executed in IMS V10.

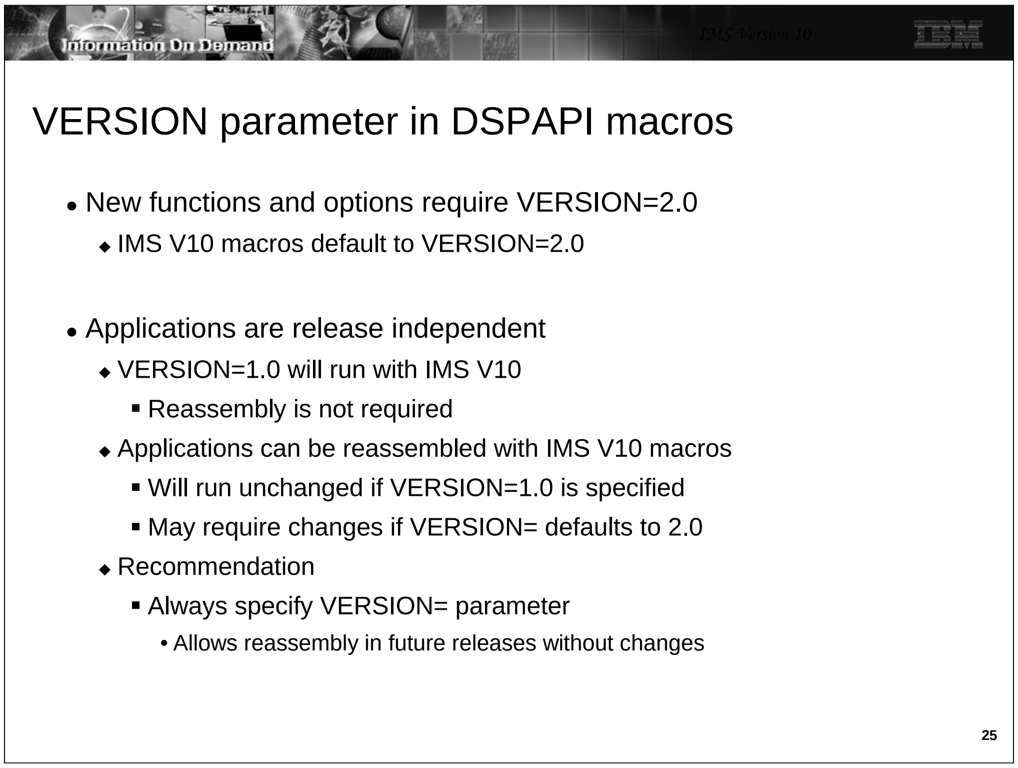


RECON Open

- RECONs may be initialized or upgraded using API
  - If VERSION=2.0 (V10) is specified on FUNC=STARTDBRC
    - STARTDBRC succeeds even with RECON not initialized or upgraded
      - RC=4 with reason code
      - Allows DBRC API program to INIT.RECON or UPGRADE
  - If VERSION=1.0 (V9) is specified on FUNC=STARTDBRC
    - STARTDBRC fails if RECON is not initialized or upgraded

24

The IMS V10 DBRC API has added the capability to open a set of RECONs which are not initialized or which are at a previous release level. This allows the application program to initialize the RECONs with an INIT.RECON command or to upgrade them with a CHANGE.RECON UPGRADE command. This could be useful for utility-like programs which create and upgrade RECONs.


VERSION parameter in DSPAPI macros

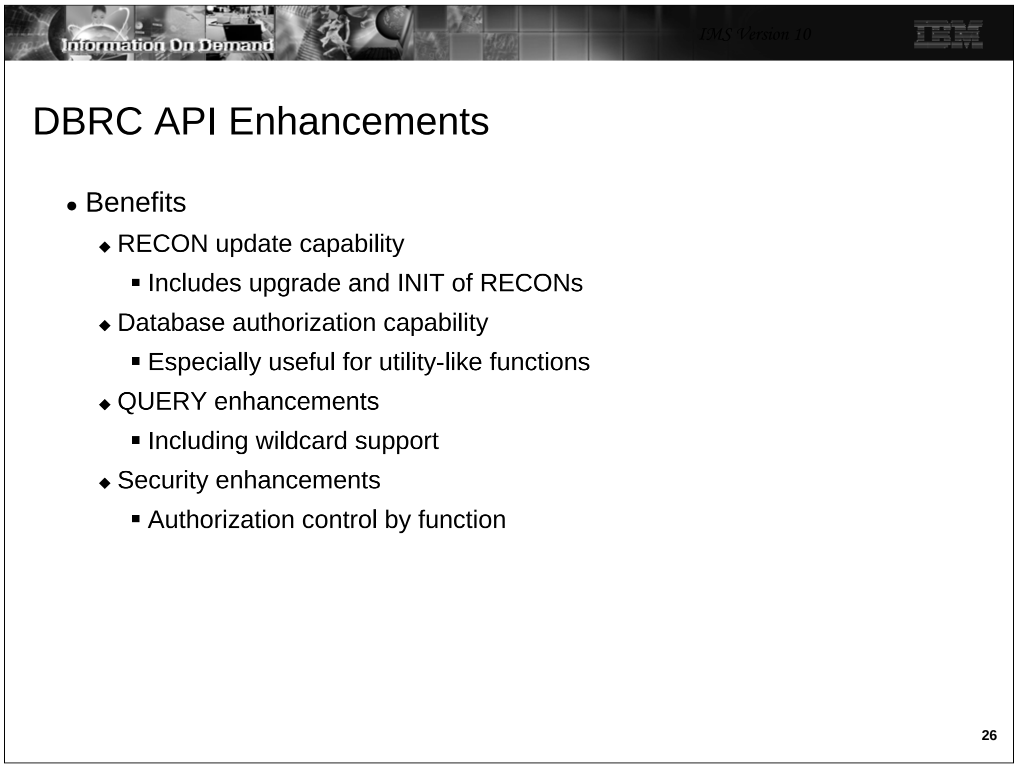
- New functions and options require VERSION=2.0
  - IMS V10 macros default to VERSION=2.0
- Applications are release independent
  - VERSION=1.0 will run with IMS V10
    - Reassembly is not required
  - Applications can be reassembled with IMS V10 macros
    - Will run unchanged if VERSION=1.0 is specified
    - May require changes if VERSION= defaults to 2.0
  - Recommendation
    - Always specify VERSION= parameter
      - Allows reassembly in future releases without changes

25

Each DBRC API macro includes the VERSION= parameter. New functions, such as AUTH, and new options, such as READONLY=YES, require VERSION=2.0. In IMS V9 the only valid value for VERSION was 1.0. It was also the default. In IMS V10 VERSION= defaults to 2.0.

Applications written in IMS V9 will continue to run in V10 without change. Reassembly is not required. In fact, reassembly could cause the program to change due to the change in the default for VERSION=. In some cases, the default of VERSION=2.0 may cause different results from the previous default of VERSION=1.0. This is not always the case. Some of the changes are only the use of previously reserved bytes in the control blocks that are produced. In any case, if you wrote a program for IMS V9 and reassemble it using an IMS V10 macro library, it is safest to specify VERSION=1.0 on the DFSAPI macros before the reassembly.

Since the VERSION= parameter defaults to the latest level of the macros and later levels may produce different results, it is safest to specify the VERSION= parameter value explicitly. This will ensure that future assemblies of DBRC API programs will produce the same results.

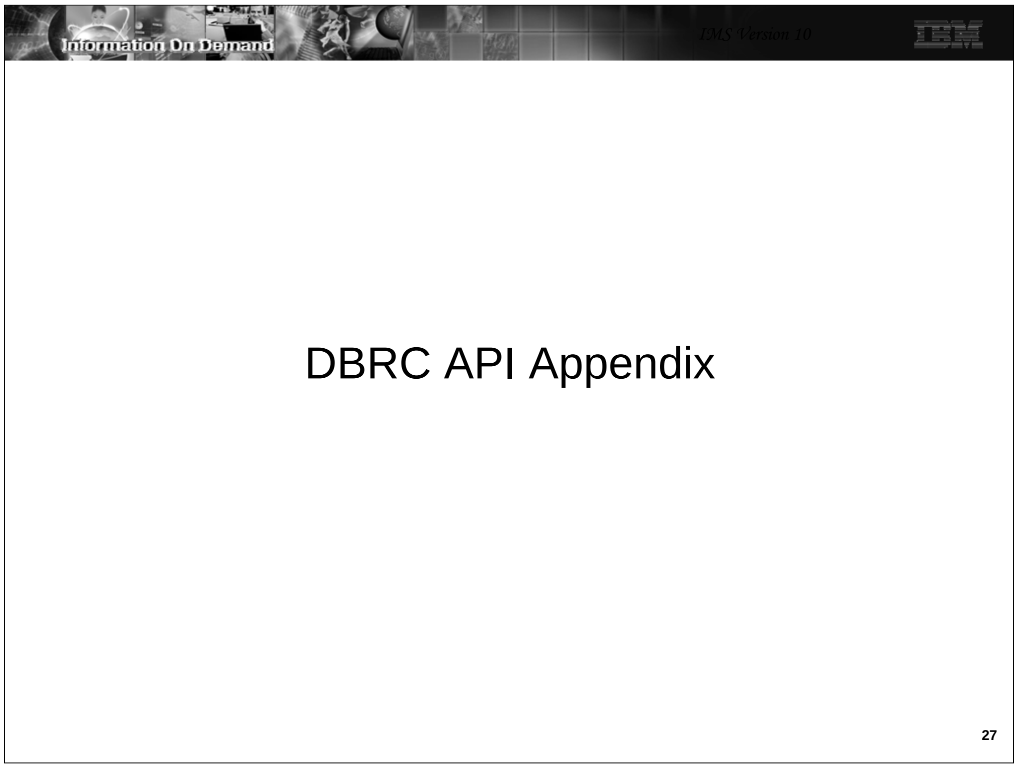


DBRC API Enhancements

- Benefits
  - RECON update capability
    - Includes upgrade and INIT of RECONs
  - Database authorization capability
    - Especially useful for utility-like functions
  - QUERY enhancements
    - Including wildcard support
  - Security enhancements
    - Authorization control by function

26

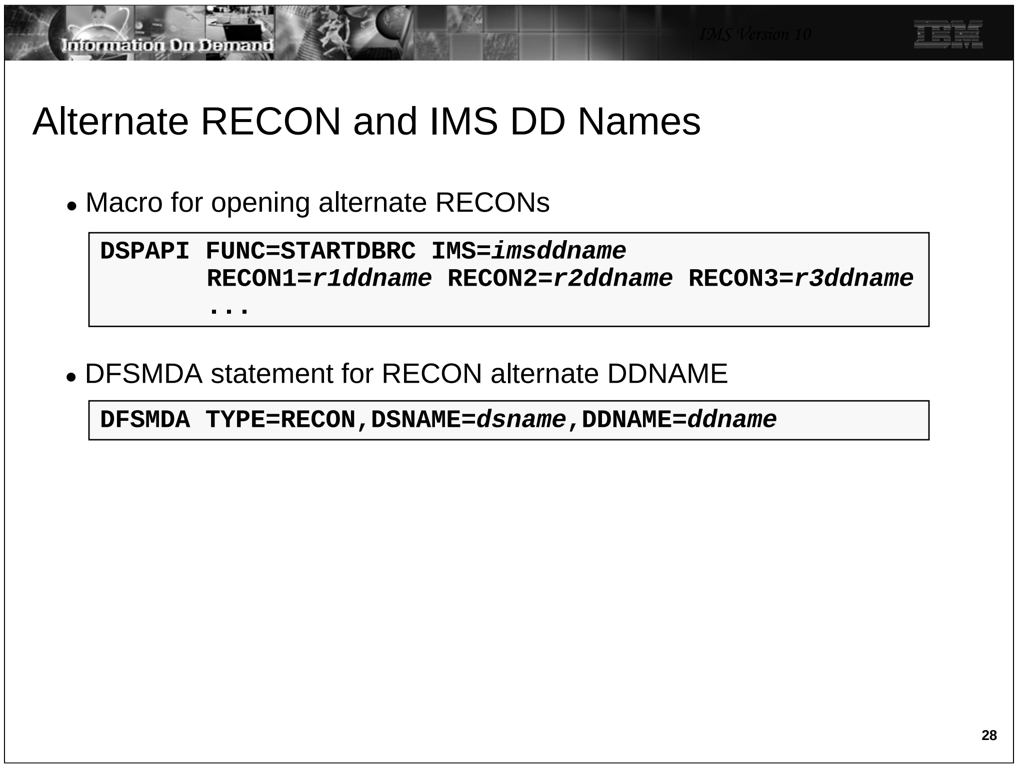
The DBRC API is described in the IMS V10 *System Programming API Reference* manual. It contains details on the DSPAPI macro, its parameters and usage, and on the control blocks created by DBRC API requests.



DBRC API Appendix

27

This appendix has more information on the DBRC API.



Information On Demand

IMS Version 10

Alternate RECON and IMS DD Names

- Macro for opening alternate RECONs

```
DSPAPI FUNC=STARTDBRC IMS=imsddname
       RECON1=r1ddname RECON2=r2ddname RECON3=r3ddname
       ...
```

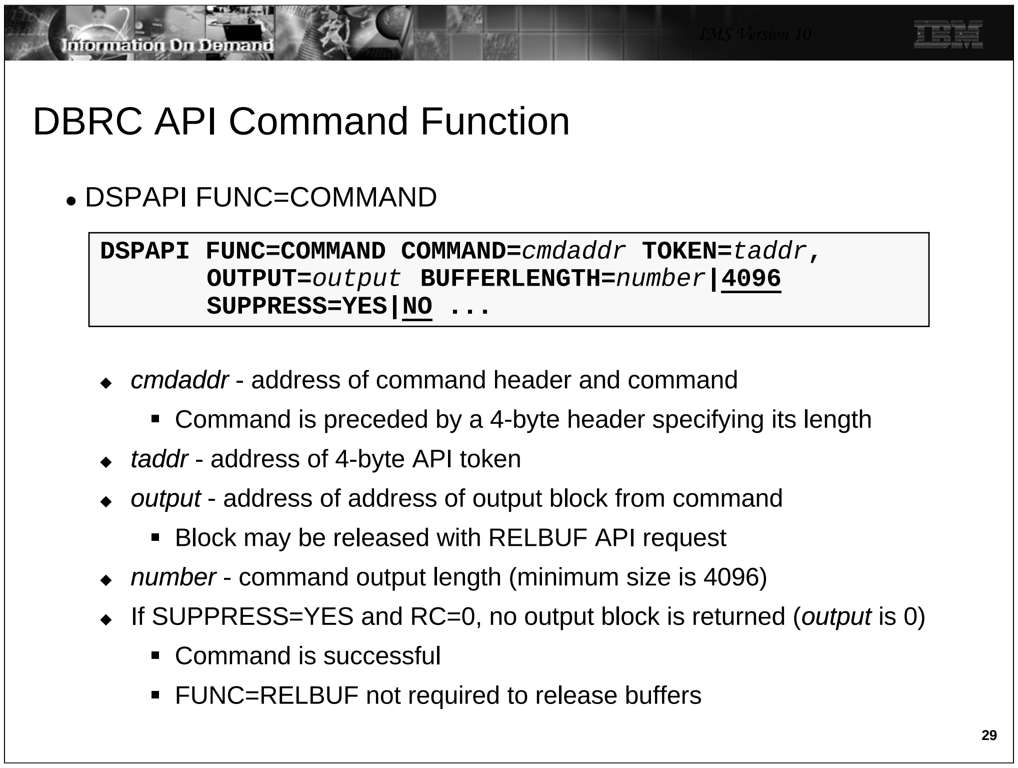
- DFSMDA statement for RECON alternate DDNAME

```
DFSMDA TYPE=RECON,DSNAME=dsname,DDNAME=ddname
```

28

The first example shows the DFSAPI FUNC=STARTDBRC macro with the parameters for specifying alternate RECON and IMS DD names.

The second example shows the DFSMDA macro to specify an alternate DD name for a RECON. A new parameter, DDNAME=, has been added to the DFSMDA macro for TYPE=RECON.



DBRC API Command Function

- DSPAPI FUNC=COMMAND

```
DSPAPI FUNC=COMMAND COMMAND=cmdaddr TOKEN=taddr,
       OUTPUT=output BUFFERLENGTH=number|4096
       SUPPRESS=YES|NO ...
```

- *cmdaddr* - address of command header and command
  - Command is preceded by a 4-byte header specifying its length
- *taddr* - address of 4-byte API token
- *output* - address of address of output block from command
  - Block may be released with RELBUF API request
- *number* - command output length (minimum size is 4096)
- If SUPPRESS=YES and RC=0, no output block is returned (*output* is 0)
  - Command is successful
  - FUNC=RELBUF not required to release buffers

29

FUNC=COMMAND is used on the DSPAPI macro to issue a DBRC command. The COMMAND= parameter specifies the address where the command resides. The command consists of a header followed by a DBRC command. The header is a full word containing the length (in bytes) of the following command. The output block address is returned at the address specified on the OUTPUT parameter. The output of the DBRC command is written in this output block. This block is shown on the next page.

The BUFFERLENGTH= parameter specifies the size of the buffer used for the output of the command. The output is written in the output block produced by the request. It is formatted for display to a terminal rather than printed to a SYSPRINT data set. The output buffer consists of:

• A halfword containing the length of the buffer which includes this halfword.

• A halfword of zeroes.

• One or more generated lines of output.

Each generated line consists of:

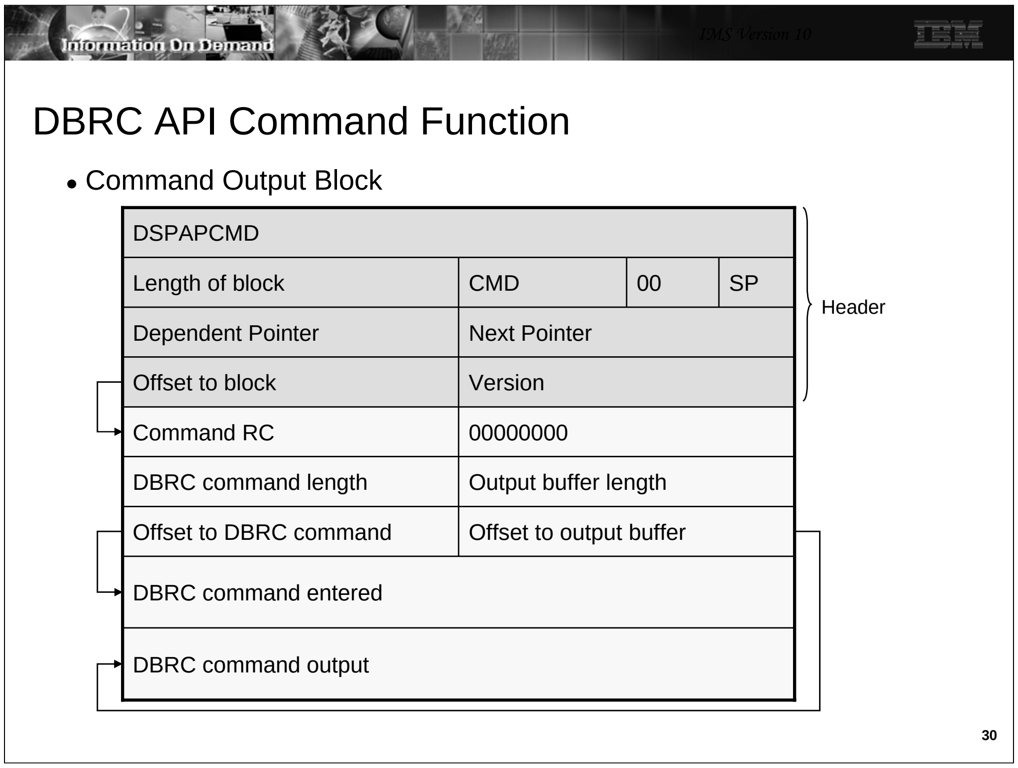
• A halfword containing the length of the line which includes this halfword.

• A halfword of zeroes.

• The generated line of output.

Carriage control characters are removed from each line. Each line is terminated with a New Line (NL) character (X'15'). Blank lines are removed from the output. The output buffer has a default length of 4,096 bytes. A buffer up to 32,760 bytes may be requested. If the buffer is not large enough to hold all the lines of output, the last line is DSP0057I MESSAGE BUFFER FULL, OUTPUT TRUNCATED.

SUPPRESS=YES may be used so that no output block is created when a return code of 0 is returned. This eliminates the possible need to release the storage for the block with a FUNC=RELBUF request. SUPPRESS=NO is the default.



DBRC API Command Function

- Command Output Block

This shows the format of the output block created by DSPAPI with FUNC=COMMAND. It has the same structure as the blocks created by DSPAPI with FUNC=QUERY. The first 32 bytes are the prefix of the block. The lines in the diagram, other than the last two line, represent 8 bytes.

• DSPAPCMD is the name of the mapping macro for this block. It is stored in the first 8 bytes of the block.

• CMD is a 2-byte indicator that this is a command block.

• 00 is one byte of x'00'.

• SP is the subpool in which this block is stored.

• The dependent pointer and next pointer are not used in the command output block. These blocks are not chained to other blocks.

• Version is the version of this block. It's value is 2.0 in IMS V10.

• Command RC is the return code from the command.

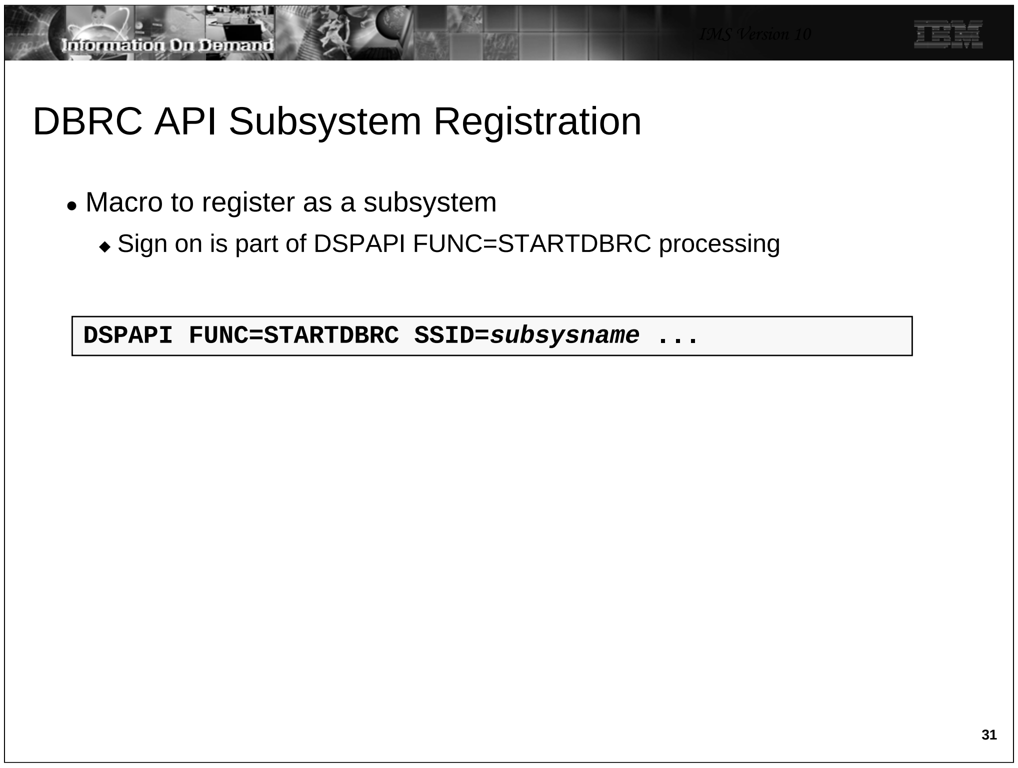
• 00000000 is four bytes of x'00'.

• DBRC command length is the length of the command passed by the DSPAPI FUNC=COMMAND macro.

• DBRC command entered is the DBRC command entered by the DSPAPI FUNC=COMMAND macro. It is copied from the address specified in the macro to this block.

•Output buffer length is the length of the output buffer that was specified in the DSPAPI FUNC=COMMAND macro.

• The offset fields are 4-byte offsets to the indicated fields in the block.



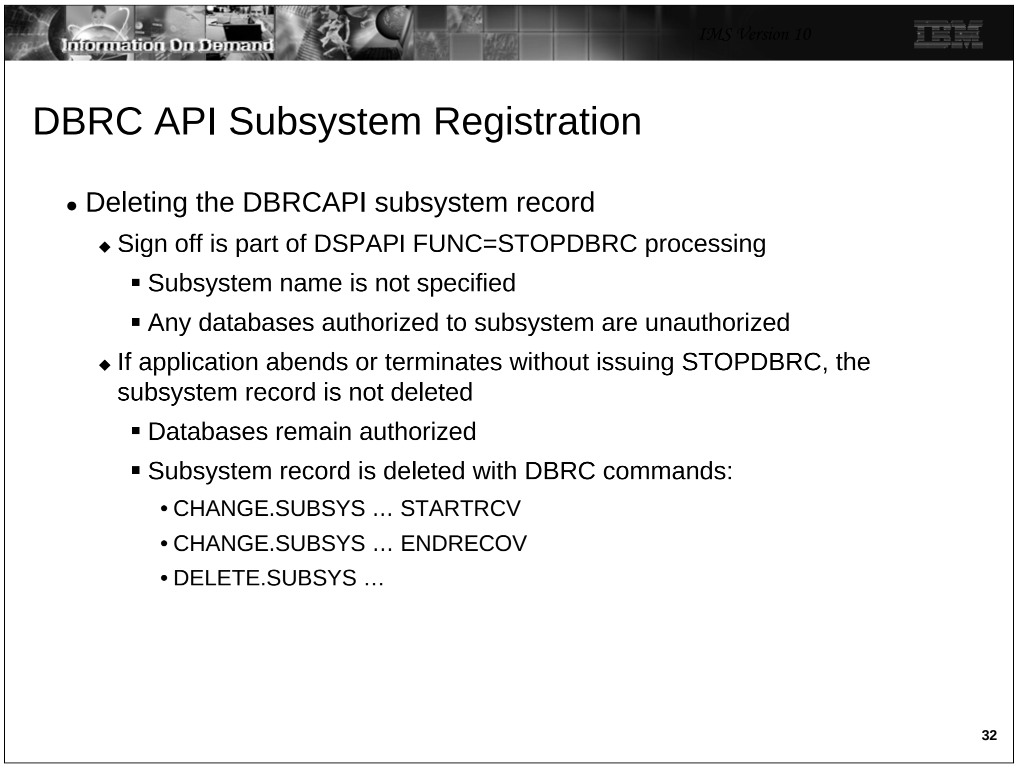
DBRC API Subsystem Registration

- Macro to register as a subsystem
  - Sign on is part of DSPAPI FUNC=STARTDBRC processing

```
DSPAPI FUNC=STARTDBRC SSID=subsysname ...
```

The STARTDBRC function is enhanced to support the creation of subsystem records. Subsystem registration is done by adding an SSID= parameter to the DSPAPI FUNC=STARTDBRC macro. The meanings of the other parameters on the macro are unchanged.

A new type of subsystem has been added to the those that are stored in the SUBSYS record. This type is DBRCAPI. It is used when DSPAPI FUNC=STARTDBRC is used to create the SUBSYS record.



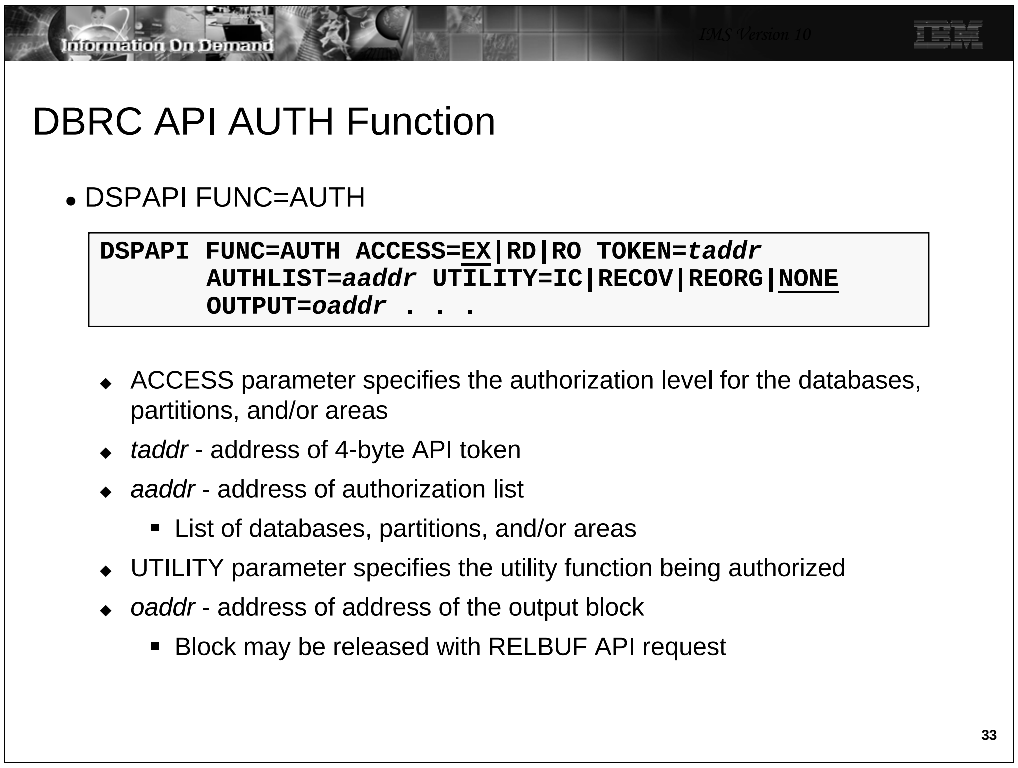
DBRC API Subsystem Registration

- Deleting the DBRCAPI subsystem record
  - Sign off is part of DSPAPI FUNC=STOPDBRC processing
    - Subsystem name is not specified
    - Any databases authorized to subsystem are unauthorized
  - If application abends or terminates without issuing STOPDBRC, the subsystem record is not deleted
    - Databases remain authorized
    - Subsystem record is deleted with DBRC commands:
      - CHANGE.SUBSYS … STARTRCV
      - CHANGE.SUBSYS … ENDRECOV
      - DELETE.SUBSYS …

32

The DBRCAPI subsystem records are deleted by DSPAPI FUNC=STOPDBRC macros. If any databases are authorized to the subsystem, they are unauthorized when the macro is processed.

If a DBRCAPI application abends or if it terminates normally without issuing the DSPAPI FUNC=STOPDBRC, its subsystem record is not deleted. Any databases authorized to the subsystem remain authorized. The subsystem record may be deleted by issuing the same commands that delete any subsystem record. If the application has authorized databases, then the CHANGE.SUBSYS STARTRCV and CHANGE.SUBSYS ENDRECOV are used to delete the authorizations. Database authorizations must be deleted before the subsystem record can be deleted. The DELETE.SUBSYS command is used to delete the subsystem record.


Information On Demand
IMS Version 10
IBM

DBRC API AUTH Function

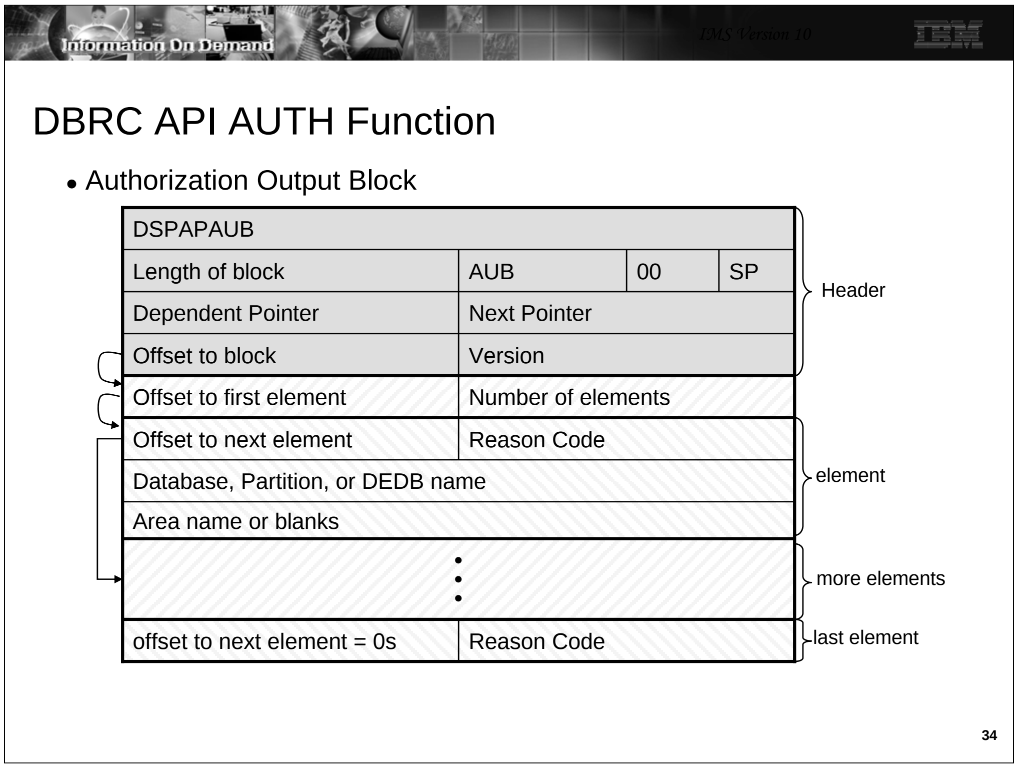
- DSPAPI FUNC=AUTH

```
DSPAPI FUNC=AUTH ACCESS=EX|RD|RO TOKEN=taddr
       AUTHLIST=aaddr UTILITY=IC|RECOV|REORG|NONE
       OUTPUT=oaddr . . .
```

- ACCESS parameter specifies the authorization level for the databases, partitions, and/or areas
- *taddr* - address of 4-byte API token
- *aaddr* - address of authorization list
  - List of databases, partitions, and/or areas
- UTILITY parameter specifies the utility function being authorized
- *oaddr* - address of address of the output block
  - Block may be released with RELBUF API request

AUTHLIST= specifies the address of the list of databases, partitions, and/or Fast Path areas to be authorized. The list consists of a fullword that contains the number of elements in the list, a fullword that contains the length of an element, followed by one or more elements. Each element consists of an 8-character database name, partition name, or Fast Path DEDB name and 8 characters of blanks (X'40') or a Fast Path area name.

The UTILITY= parameter is designed to be used by applications that perform utility functions. UTILITY= values other than NONE specify that DBRC is to evaluate the authorization request as it would a request from the indicated utility. Utility authorization requests bypass some of the database and data set status indicators. Image Copy (IC) can get authorization when the Image Copy Needed or Prevent Further Authorization flags are on. Reorganization (REORG) can get authorization when the Image Copy Needed, Prevent Further Authorization, or Read Only flags are on. Recovery (RECOV) can get authorization when any of the status flags are on. These include the Image Copy Needed, Prevent Further Authorization, Read Only, Recovery Needed, and Backout Needed flags.



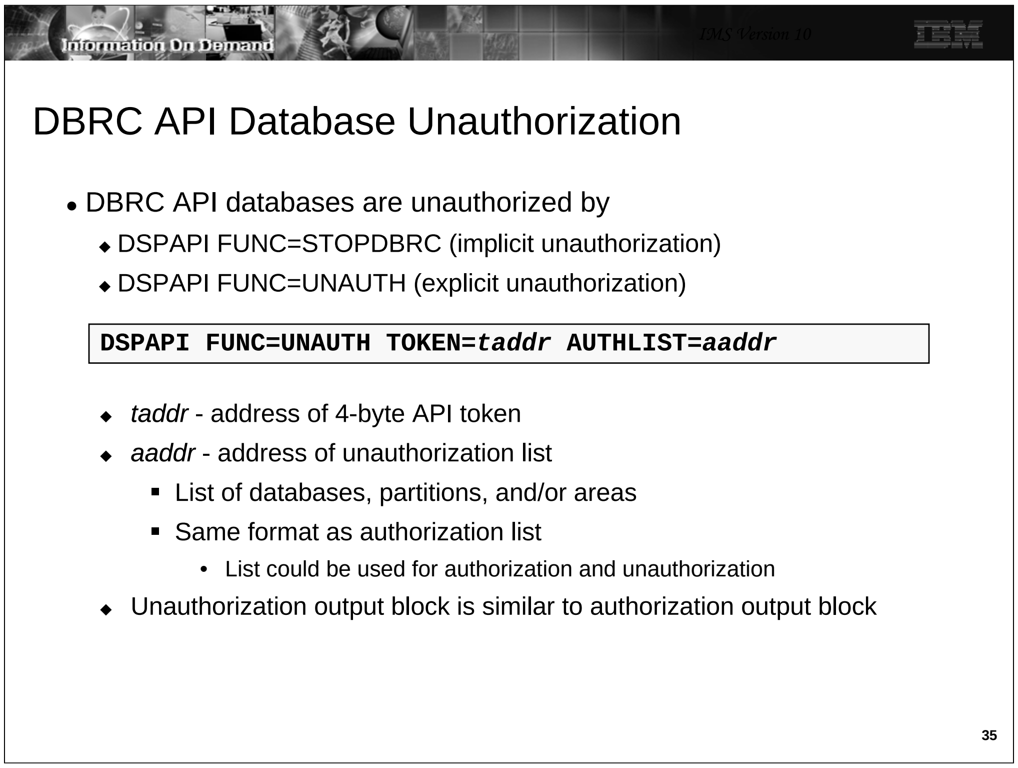
DBRC API AUTH Function

- Authorization Output Block

DSPAPAUB			
Length of block	AUB	00	SP
Dependent Pointer	Next Pointer		
Offset to block	Version		
Offset to first element	Number of elements		
Offset to next element	Reason Code		
Database, Partition, or DEDB name			
Area name or blanks			
• • •			
offset to next element = 0s	Reason Code		

This shows the format of the output block created by DSPAPI with FUNC=AUTH. The prefix is like the prefix of other blocks created by DSPAPI. The first 32 bytes are the prefix of the block.

- DSPAPAUB is the name of the mapping macro for this block. It is stored in the first 8 bytes of the block.
- AUB is a 2-byte indicator that this is an authorization block.
- 00 is one byte of x’00’.
- SP is the subpool in which this block is stored.
- The dependent pointer is not used in the authorization block. The next pointer points to the next authorization block if one exists.
- The offset to block is the pointer to the data after the prefix.
- Version is the version of this block. It’s value is 2.0 in IMS V10.
- Offset to first element points to the first element.
- Number of elements is the number of elements stored in this block.
- Each element has four fields.
    - The first field is a pointer to the next element. The last element has zeros in this pointer.
    - The reason code is the indication of the success of the authorization attempt for this database, partition, or area.
    - The name of the database or partition is in the next field.
    - For Fast Path areas, the area name is in the next field. For full function databases, including HALDB partitions, this field is blanks.



DBRC API Database Unauthorization

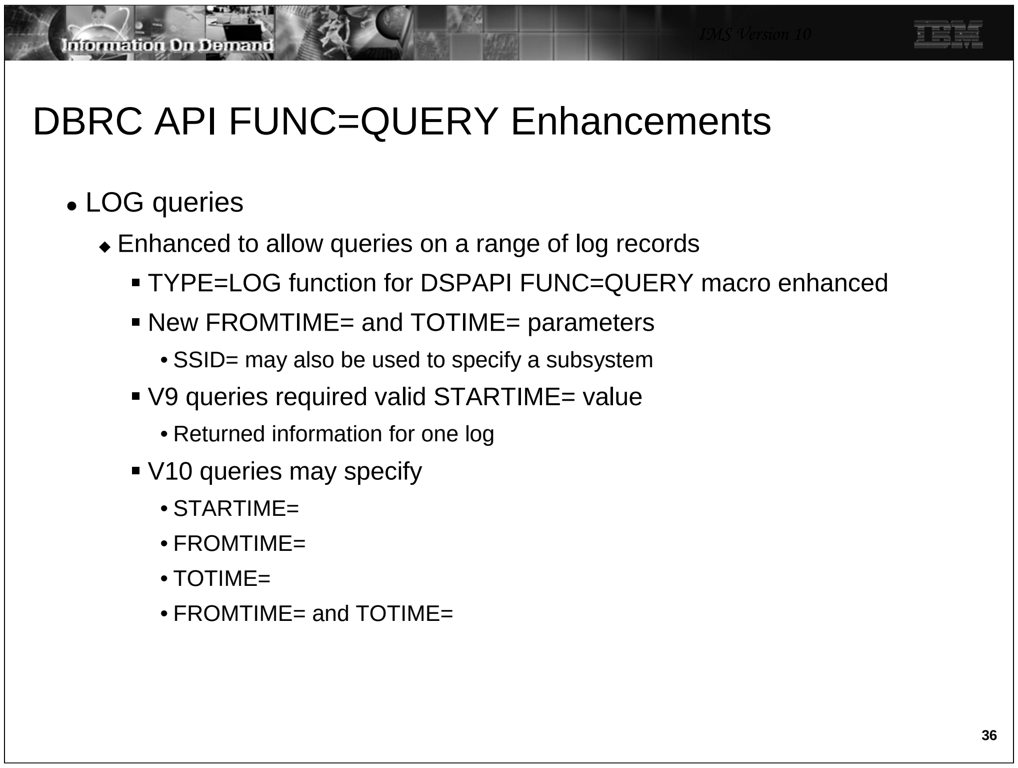
- DBRC API databases are unauthorized by
  - DSPAPI FUNC=STOPDBRC (implicit unauthorization)
  - DSPAPI FUNC=UNAUTH (explicit unauthorization)

```
DSPAPI FUNC=UNAUTH TOKEN=taddr AUTHLIST=aaddr
```

- *taddr* - address of 4-byte API token
- *aaddr* - address of unauthorization list
  - List of databases, partitions, and/or areas
  - Same format as authorization list
    - List could be used for authorization and unauthorization
- Unauthorization output block is similar to authorization output block

35

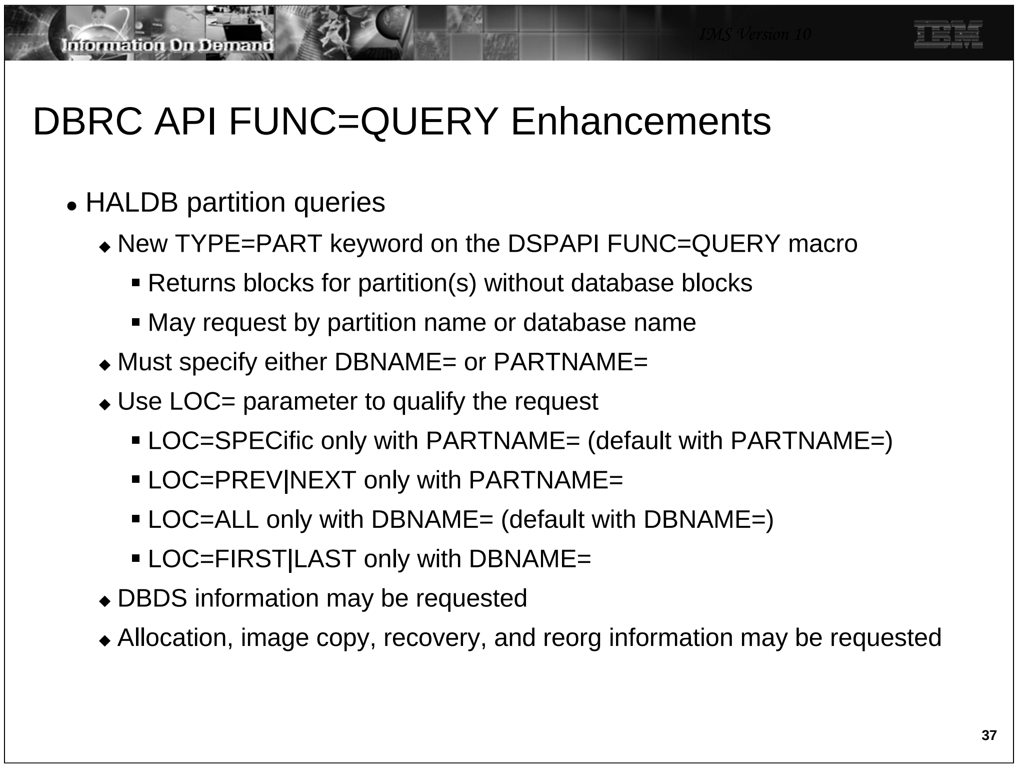
Application programs which authorize databases can also unauthorize them. The STOPDBRC function unauthorizes all of the databases that were authorized. Individual databases may be unauthorized explicitly by using FUNC=UNAUTH.


Information On Demand
IMS Version 10

DBRC API FUNC=QUERY Enhancements

- LOG queries
  - Enhanced to allow queries on a range of log records
    - TYPE=LOG function for DSPAPI FUNC=QUERY macro enhanced
    - New FROMTIME= and TOTIME= parameters
      - SSID= may also be used to specify a subsystem
    - V9 queries required valid STARTIME= value
      - Returned information for one log
    - V10 queries may specify
      - STARTIME=
      - FROMTIME=
      - TOTIME=
      - FROMTIME= and TOTIME=

The IMS V9 implementation of FUNC=QUERY TYPE=LOG has been enhanced in V10. QUERY requests for TYPE=LOG required a STARTIME= parameter. This specified the start time of a log. V10 adds the capability to request information about a range of logs. If FROMTIME= is specified without TOTIME=, all logs from the specified time to the present are returned. If TOTIME= is specified without FROMTIME=, all logs up to the specified time are returned. If both FROMTIME= and TOTIME= are specified, all logs within the time range are returned. Since information about multiple logs may be returned with these time range requests, the LOGINFO blocks (DSPAPQLI) may be chained with their “next” pointers.



IMS Version 10

DBRC API FUNC=QUERY Enhancements

- HALDB partition queries
  - New TYPE=PART keyword on the DSPAPI FUNC=QUERY macro
    - Returns blocks for partition(s) without database blocks
    - May request by partition name or database name
  - Must specify either DBNAME= or PARTNAME=
  - Use LOC= parameter to qualify the request
    - LOC=SPECific only with PARTNAME= (default with PARTNAME=)
    - LOC=PREV|NEXT only with PARTNAME=
    - LOC=ALL only with DBNAME= (default with DBNAME=)
    - LOC=FIRST|LAST only with DBNAME=
  - DBDS information may be requested
  - Allocation, image copy, recovery, and reorg information may be requested

37

The IMS V9 implementation of FUNC=QUERY TYPE=DB could be used to get information about HALDB partitions. This was done by specifying a database name. Information about all of the partitions was returned. IMS V10 adds FUNC=QUERY TYPE=PART. This allows you to get information about a partition without specifying the database name. It also allows you to retrieve partition information without retrieving the database information.

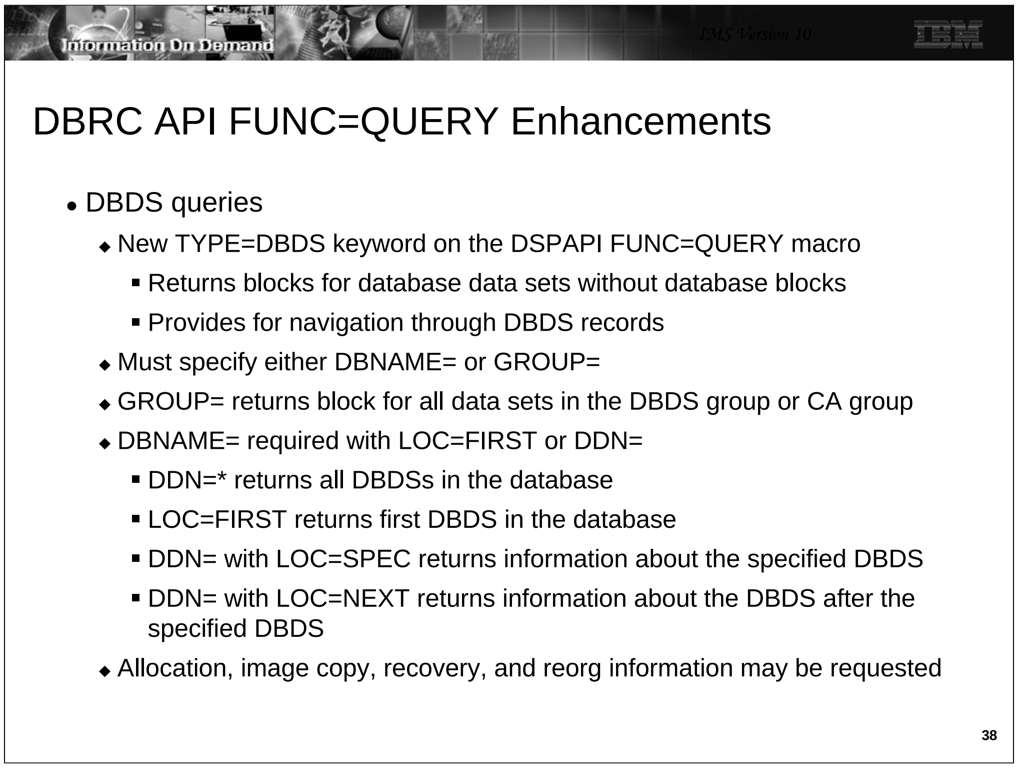
Either the PARTNAME= or DBNAME= parameter must be specified. Both of these parameters may not be specified on the same macro.

When you specify PARTNAME= you may request the a specific partition by name. Alternatively, you may request the next or previous partition relative to the partition specified.

When you specify DBNAME= you may request all of the partitions in the database. Alternatively, you may request the first or last partition in the database.

As with FUNC=QUERY TYPE=DB you may request DBDS information for the partitions returned and you may request allocation (ALLOC), image copy (IC), recovery (RECOV), and reorganization (REORG) information for these database data sets.

Partitions in a HALDB database may be disabled. This is done with a CHANGE.PART command with the DISABLE keyword. Definitions for disabled partitions remain in the RECONs, but they are not actively used. That is, they are no longer part of the definition of their master database. Disabled partitions are not returned by FUNC=QUERY TYPE=PART macros where LOC=ALL, FIRST, LAST, PREV, or NEXT is specified. They can be returned with PARTNAME= LOC=SPEC.



DBRC API FUNC=QUERY Enhancements

- DBDS queries
  - New TYPE=DBDS keyword on the DSPAPI FUNC=QUERY macro
    - Returns blocks for database data sets without database blocks
    - Provides for navigation through DBDS records
  - Must specify either DBNAME= or GROUP=
  - GROUP= returns block for all data sets in the DBDS group or CA group
  - DBNAME= required with LOC=FIRST or DDN=
    - DDN=* returns all DBDSs in the database
    - LOC=FIRST returns first DBDS in the database
    - DDN= with LOC=SPEC returns information about the specified DBDS
    - DDN= with LOC=NEXT returns information about the DBDS after the specified DBDS
  - Allocation, image copy, recovery, and reorg information may be requested

38

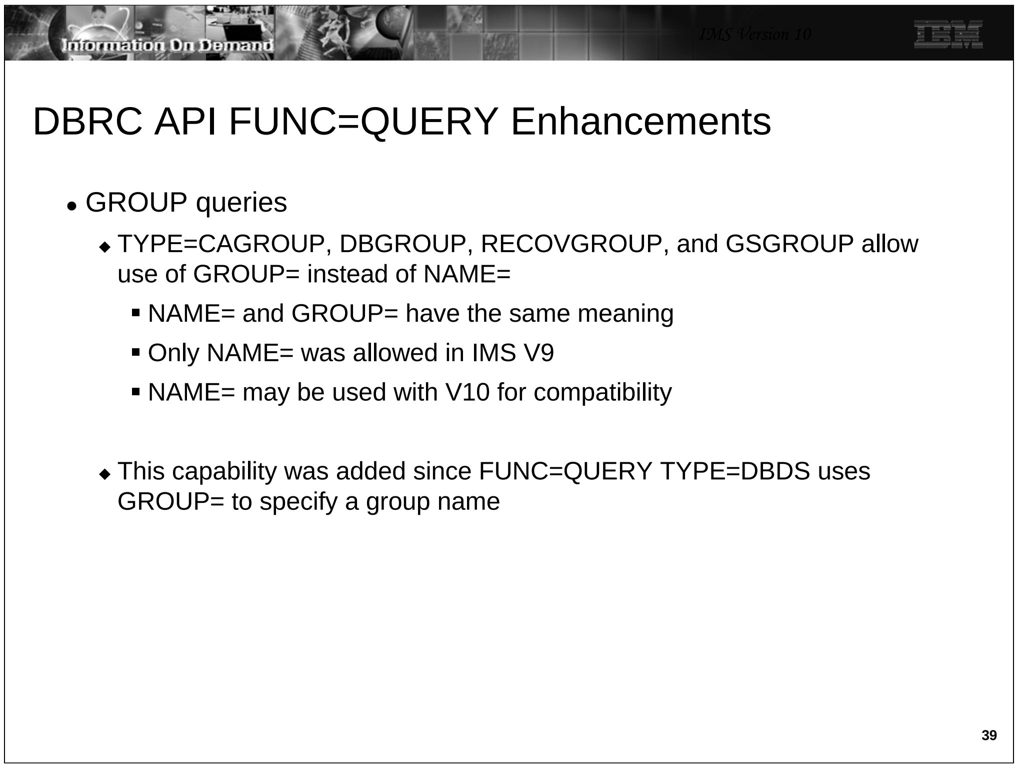
The IMS V9 implementation of FUNC=QUERY TYPE=DB could be used to get information about database data sets. This was done by specifying a database name and requesting DBDS information. Information about a specific data set in the database, all data sets in the database, or none of them could be requested. IMS V10 adds FUNC=QUERY TYPE=DBDS. This provides an alternate way of getting this information. It also allows you to retrieve information about the first DBDS in a database with one request and the next DBDSs with subsequent requests.

Either the GROUP= or DBNAME= parameter must be specified. Both of these parameters may not be specified on the same macro.

When you specify GROUP= information about all of the database data sets in the group is returned. A group is either a DBDS group or a CA group. You cannot specify a database group or a recovery group.

When you specify DBNAME= the information returned is determined by the DDN= and LOC= parameters. If DDN=* is specified, blocks for all data sets in the database are returned. DDN=* is the default. If LOC=FIRST is specified without DDN=, the first data set in the database is returned. If DDN= is specified with LOC=SPEC, a block for the indicated data set is returned. If DDN= is specified with LOC=NEXT, the dataset after the specified data set is returned.

As with FUNC=QUERY TYPE=DB you may request allocation (ALLOC), image copy (IC), recovery (RECOV), and reorganization (REORG) information for these database data sets.

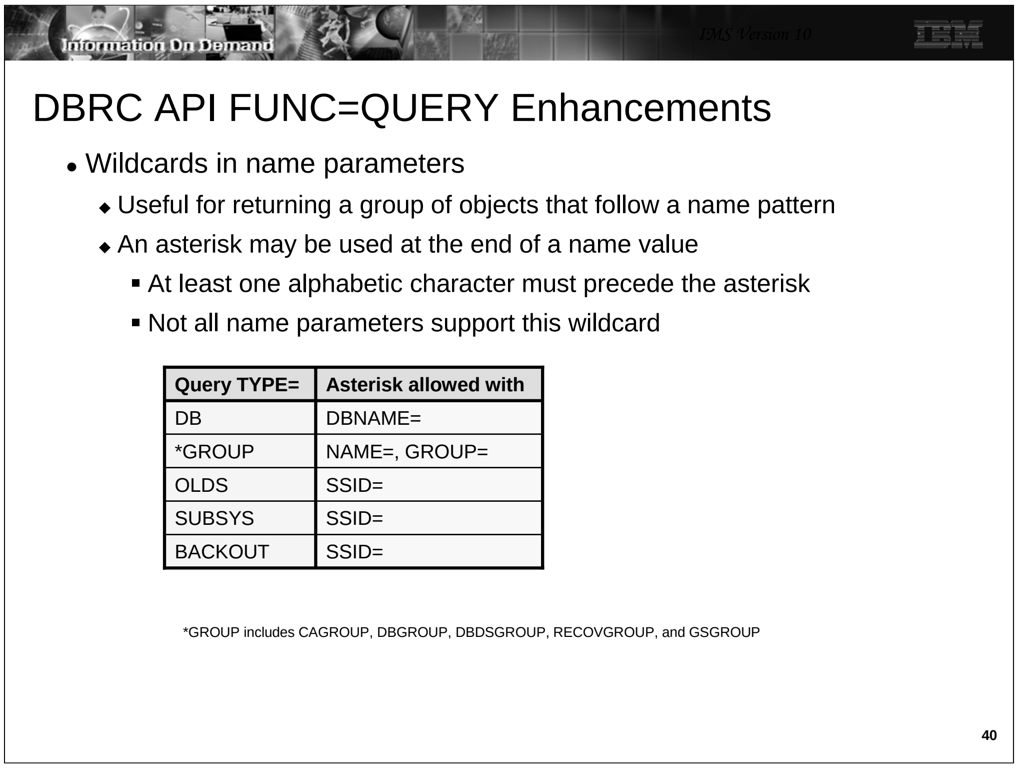


DBRC API FUNC=QUERY Enhancements

- GROUP queries
  - TYPE=CAGROUP, DBGROUP, RECOVGROUP, and GSGROUP allow use of GROUP= instead of NAME=
    - NAME= and GROUP= have the same meaning
    - Only NAME= was allowed in IMS V9
    - NAME= may be used with V10 for compatibility
  - This capability was added since FUNC=QUERY TYPE=DBDS uses GROUP= to specify a group name

39

The IMS V9 FUNC=QUERY with a TYPE= value of CAGROUP, DBGROUP, RECOVGROUP, or GSGROUP used the NAME= parameter to specify the name of the group. IMS V10 allows the use of the new GROUP= parameter to specify the name of the group. This produces exactly the same results as the use of NAME=. The use of GROUP= was added to be compatible with FUNC=QUERY TYPE=DBDS where GROUP= is used for the name of the group. NAME= is still supported when TYPE= specifies a group.



DBRC API FUNC=QUERY Enhancements

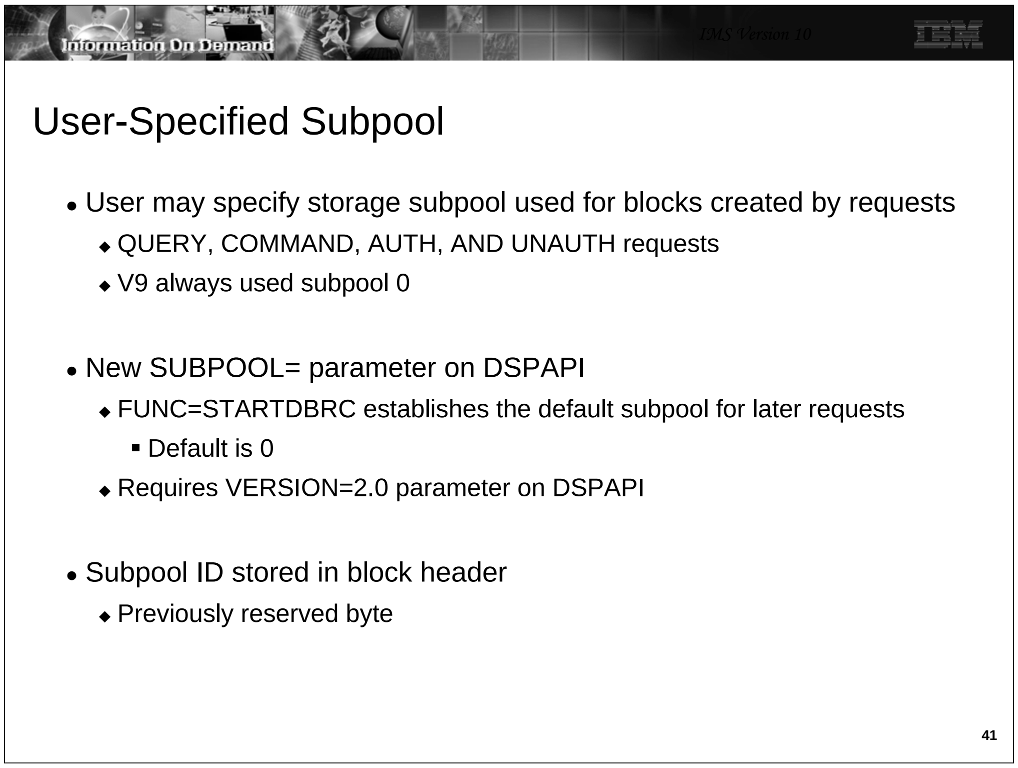
- Wildcards in name parameters
  - Useful for returning a group of objects that follow a name pattern
  - An asterisk may be used at the end of a name value
    - At least one alphabetic character must precede the asterisk
    - Not all name parameters support this wildcard

Query TYPE=	Asterisk allowed with
DB	DBNAME=
*GROUP	NAME=, GROUP=
OLDS	SSID=
SUBSYS	SSID=
BACKOUT	SSID=

*GROUP includes CAGROUP, DBGROUP, DBDSGROUP, RECOVGROUP, and GSGROUP

V10 has added a capability to use an asterisk as a wildcard in some parameters on DSPAPI TYPE=QUERY macros. This can be especially useful when a naming pattern is used for databases, subsystems, or groups. The asterisk can only be used at the end of parameters with names as values. At least one alphabetic character must precede the asterisk. Some parameters accept this wild card. Some do not.

The table shows for which parameters the asterisk is valid. The first column indicates the type of request. The row with "*GROUP" in the first column applies to QUERY requests with TYPE= values of CAGROUP, DBGROUP, DBDSGROUP, RECOVGROUP, and GSGROUP. The parameters which require names and that accept the asterisk are listed in the "Asterisk allowed with" column.


Information On Demand
IMS Version 10

User-Specified Subpool

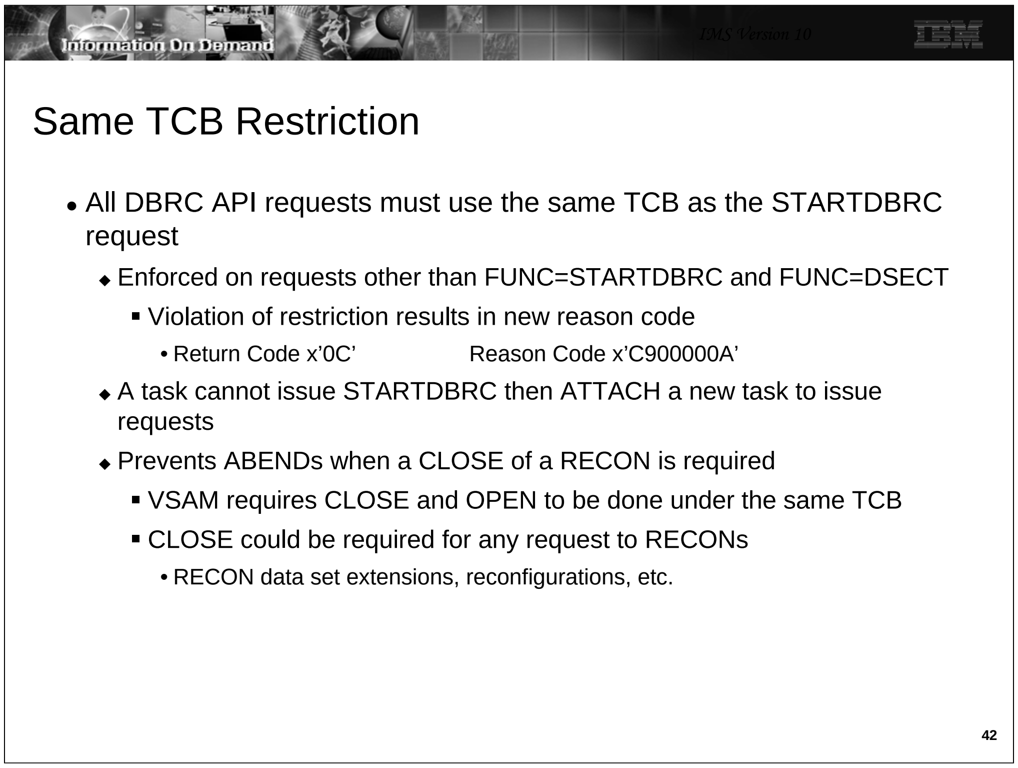
- User may specify storage subpool used for blocks created by requests
  - QUERY, COMMAND, AUTH, AND UNAUTH requests
  - V9 always used subpool 0
- New SUBPOOL= parameter on DSPAPI
  - FUNC=STARTDBRC establishes the default subpool for later requests
    - Default is 0
  - Requires VERSION=2.0 parameter on DSPAPI
- Subpool ID stored in block header
  - Previously reserved byte

41

The new SUBPOOL= parameter may be used on DSPAPI macros which create blocks. It determines which subpool will be used for the block. This new parameter is only available when VERSION=2.0 is also specified on the DSPAPI macro. The FUNC=STARTDBRC request establishes the default for the SUBPOOL= parameter on later requests. If it is not specified on the FUNC=STARTDBRC request, the default is subpool 0.

The header of blocks has been enhanced to include the subpool ID. It is stored in a previously reserved byte of the header.

Users should see the *z/OS Authorized MVS Assembler Services Guide* for valid subpools for their programs.

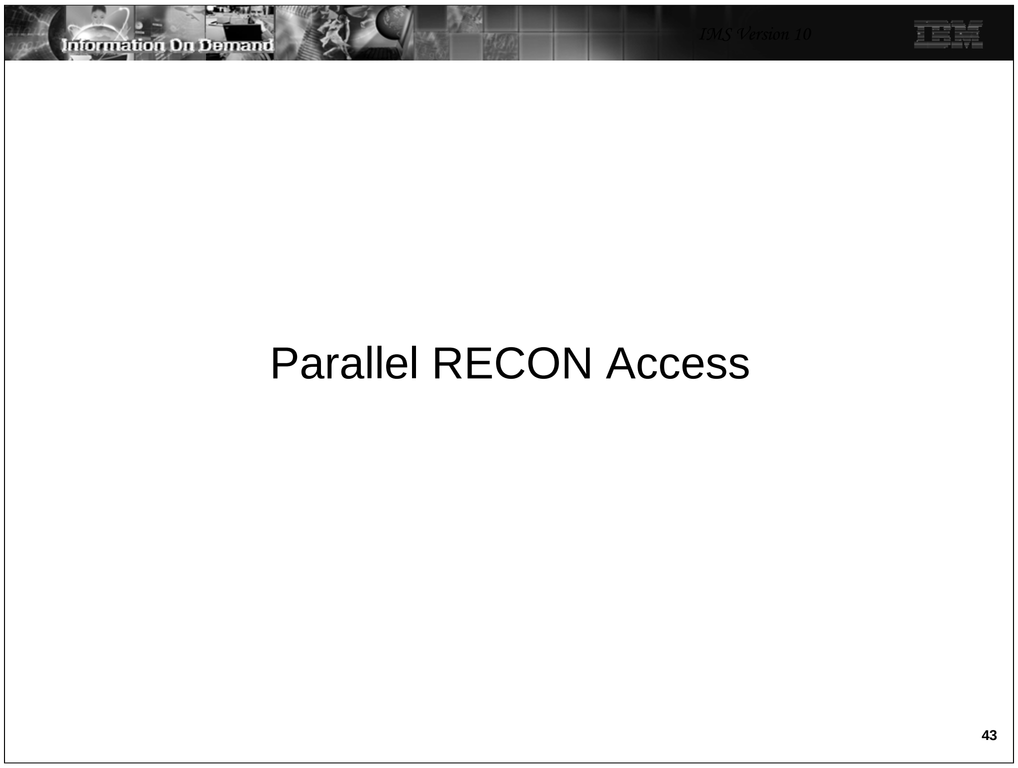


Same TCB Restriction

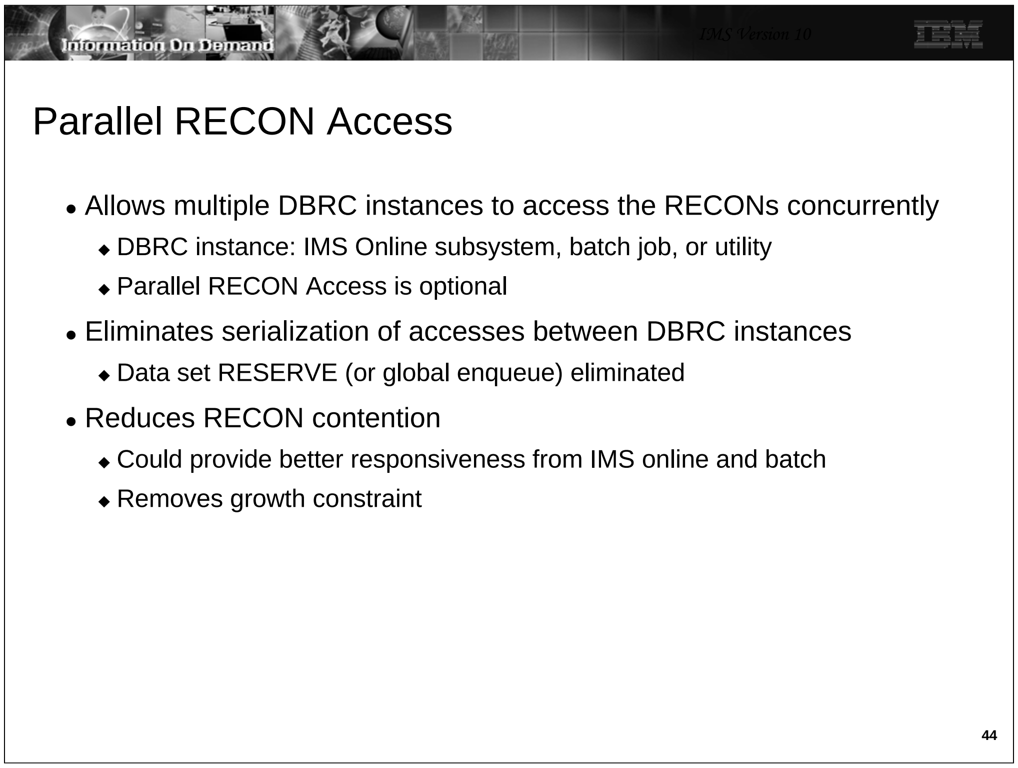
- All DBRC API requests must use the same TCB as the STARTDBRC request
  - Enforced on requests other than FUNC=STARTDBRC and FUNC=DSECT
    - Violation of restriction results in new reason code
      - Return Code x'0C' Reason Code x'C900000A'
  - A task cannot issue STARTDBRC then ATTACH a new task to issue requests
  - Prevents ABENDs when a CLOSE of a RECON is required
    - VSAM requires CLOSE and OPEN to be done under the same TCB
    - CLOSE could be required for any request to RECONs
      - RECON data set extensions, reconfigurations, etc.

IMS V10 enforces the "same TCB" restriction. All DBRC API requests from a STARTDBRC through a STOPDBRC request, must be done under the same TCB. An attempt to make such a request under a different TCB will result in a failure with return code x'0C' and reason code x'C900000A'. This is a new reason code in V10.

This enforcement is done to avoid a possible ABEND. VSAM requires that a close of a data set be done under the same TCB as the open was done. Closes under different TCBs result in ABENDs. Any access to the RECONs might require a close. One example occurs when a RECON data set is extended. If an instance of DBRC accesses the RECONs and determines that a RECON data set has new extents, it closes and reopens the data set to get the new extent information. Another example occurs when a DBRC instance determines that the RECONs have been reconfigured. That is, a RECON data set has been lost and the old spare RECON is now one of he active RECONs. This DBRC instance closes the lost RECON.



Parallel RECON Access

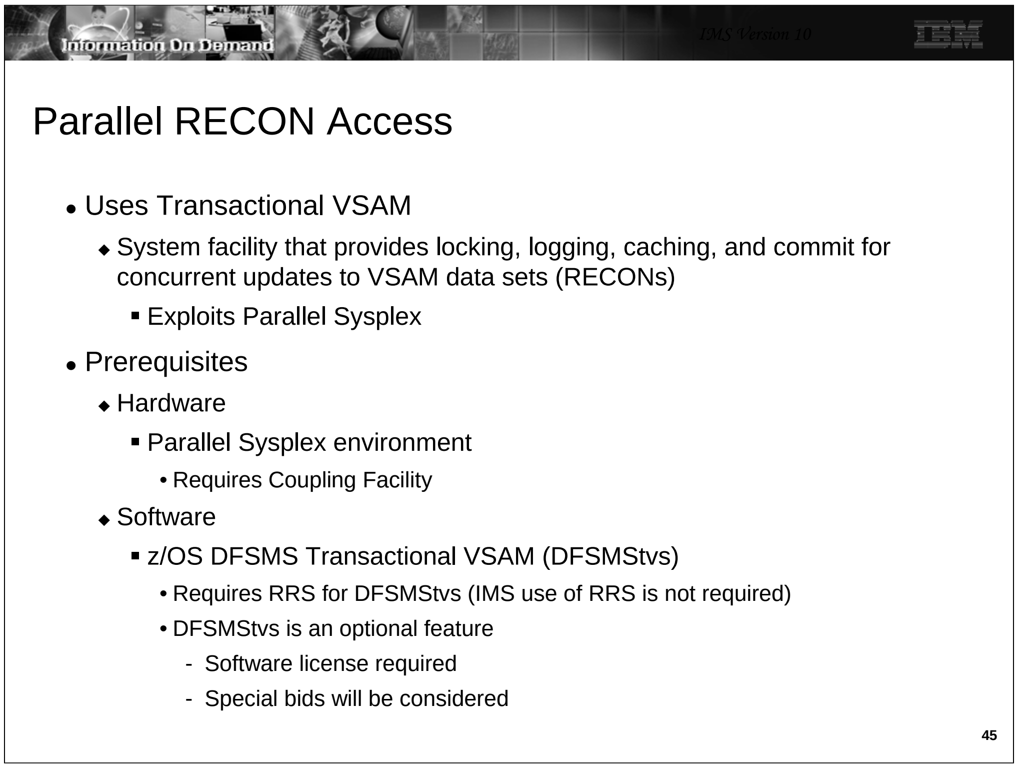


Parallel RECON Access

- Allows multiple DBRC instances to access the RECONs concurrently
  - DBRC instance: IMS Online subsystem, batch job, or utility
  - Parallel RECON Access is optional
- Eliminates serialization of accesses between DBRC instances
  - Data set RESERVE (or global enqueue) eliminated
- Reduces RECON contention
  - Could provide better responsiveness from IMS online and batch
  - Removes growth constraint

Parallel RECON Access (PRA) is a new optional capability in IMS V10. It allows multiple DBRCs to access the RECONs at the same time.

Without PRA each DBRC instance must take its turn in accessing the RECONs. Multiple DBRCs may have the RECONs open at the same time, but only one may do I/O to the RECONs at any time. This restriction is eliminated with PRA. Serial access uses RESERVEs or global enqueues to serialize access to the RECONs. PRA eliminates this serialization. This reduces contention for the RECONs. It could provide better responsiveness, especially in situations where multiple online, batch, or utility executions of IMS are doing many I/Os to the RECONs.


Information On Demand
IMS Version 10

Parallel RECON Access

- Uses Transactional VSAM
  - System facility that provides locking, logging, caching, and commit for concurrent updates to VSAM data sets (RECONs)
    - Exploits Parallel Sysplex
- Prerequisites
  - Hardware
    - Parallel Sysplex environment
      - Requires Coupling Facility
  - Software
    - z/OS DFSMS Transactional VSAM (DFSMStvs)
      - Requires RRS for DFSMStvs (IMS use of RRS is not required)
      - DFSMStvs is an optional feature
        - Software license required
        - Special bids will be considered

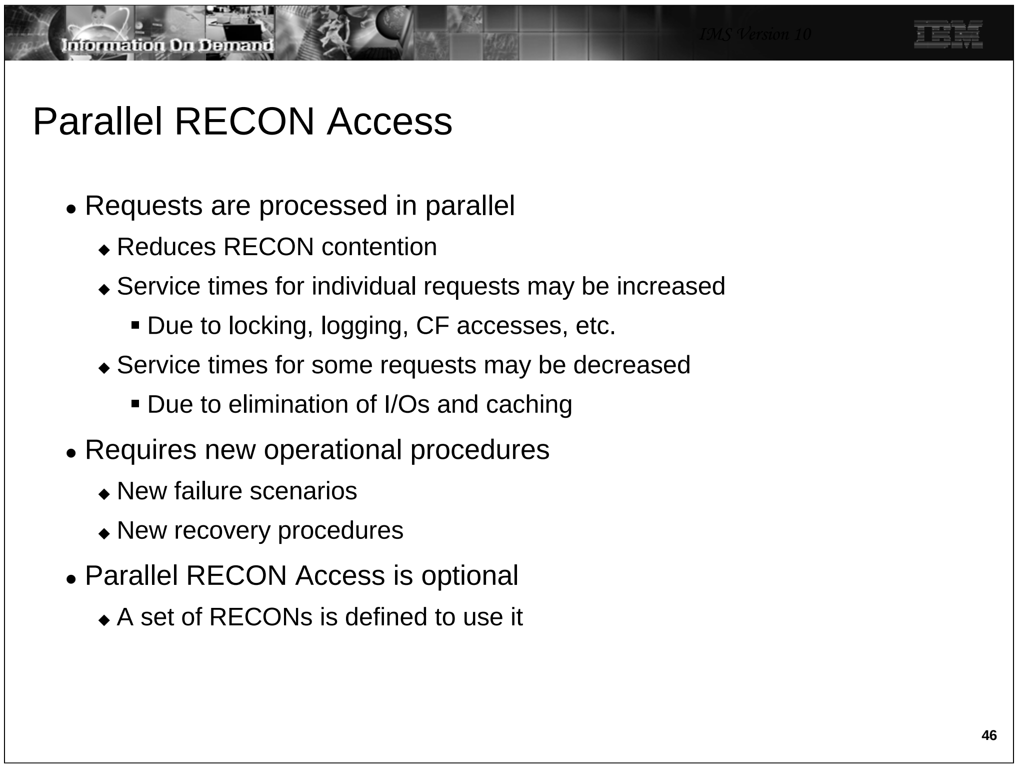
45

PRA uses Transactional VSAM. This is a system facility that is provided by DFSMS. Transactional VSAM provides locking, logging, caching, and commit coordination for VSAM data sets such as the RECONs. It requires and exploits Parallel Sysplex.

PRA requires a Parallel Sysplex including a Coupling Facility. This is true even when all of the DBRC instances are running in only one z/OS system.

PRA requires DFSMStvs. This is an optional feature of DFSMS. There is a licensing fee associated with this feature. Special bids will be considered for IMS customers using the Parallel RECON Access function, who do not already have DFSMStvs, to acquire DFSMStvs for use restricted to IMS.

DFSMStvs uses Resource Recovery Services (RRS). RRS is used for commit coordination. IMS use of RRS is not required. That is, the RRS=Y IMS execution parameter is not required.



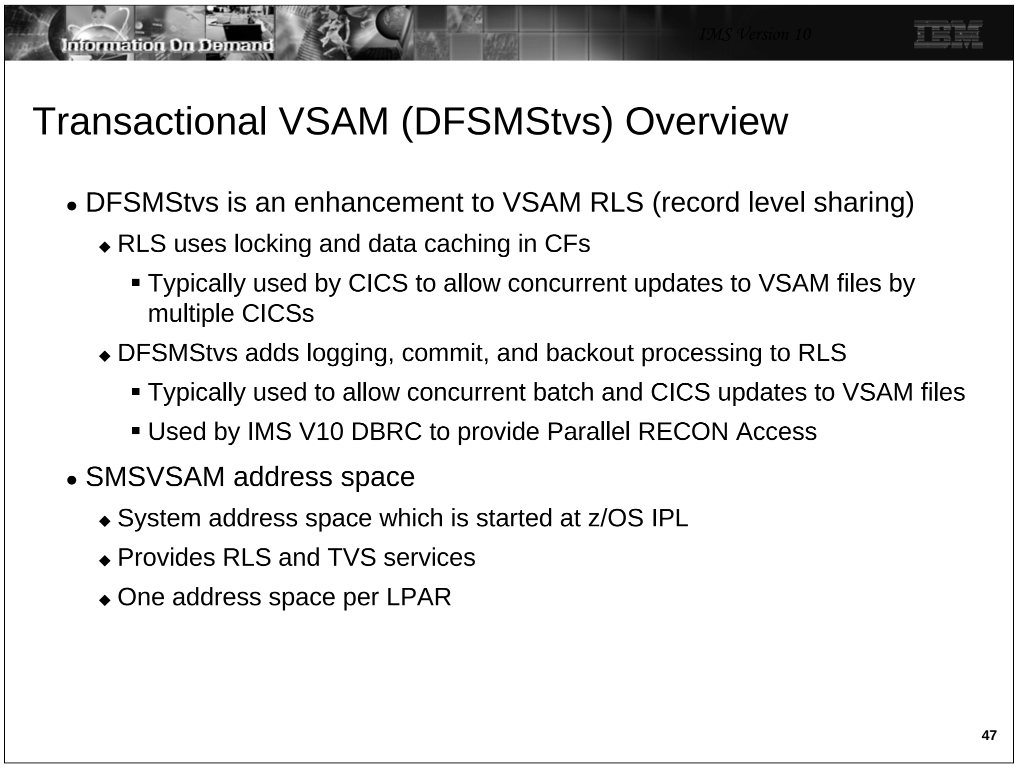
Parallel RECON Access

- Requests are processed in parallel
  - Reduces RECON contention
  - Service times for individual requests may be increased
    - Due to locking, logging, CF accesses, etc.
  - Service times for some requests may be decreased
    - Due to elimination of I/Os and caching
- Requires new operational procedures
  - New failure scenarios
  - New recovery procedures
- Parallel RECON Access is optional
  - A set of RECONs is defined to use it

Since PRA allows RECON requests to be processed in parallel, it potentially reduces RECON contention. On the other hand, service times for individual requests may increase due to the overhead of locking, logging, and Coupling Facility accesses that are required. Some services times may be decreased. This is due to the caching of RECON information in buffer pools and the CF.

PRA requires new operational procedures for recoveries. There are new possibilities for failures when the DFSMStvs environment is used.

PRA is optional in IMS V10. It is specified for a set of RECONs. There is a DBRC command for specifying PRA for a set or RECONs. Some RECON sets may use PRA while others continue to use serial access.


Information On Demand
IMS Version 10

Transactional VSAM (DFSMStvs) Overview

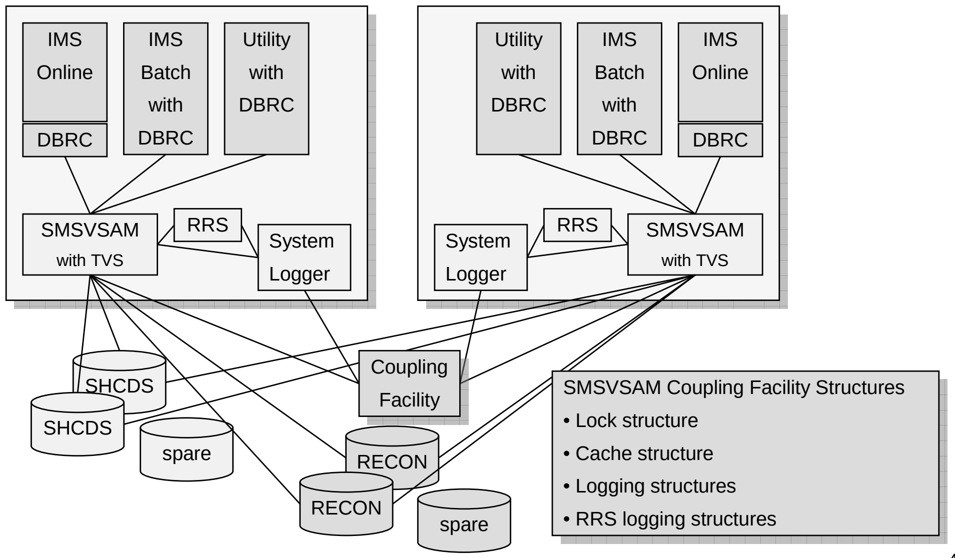
- DFSMStvs is an enhancement to VSAM RLS (record level sharing)
  - RLS uses locking and data caching in CFs
    - Typically used by CICS to allow concurrent updates to VSAM files by multiple CICSs
  - DFSMStvs adds logging, commit, and backout processing to RLS
    - Typically used to allow concurrent batch and CICS updates to VSAM files
    - Used by IMS V10 DBRC to provide Parallel RECON Access
- SMSVSAM address space
  - System address space which is started at z/OS IPL
  - Provides RLS and TVS services
  - One address space per LPAR

47

DFSMStvs (TVS) is based on VSAM Record Level Sharing (RLS). RLS uses locking and data caching to provide data sharing capabilities for VSAM files used by CICS systems. It allows multiple CICS systems to concurrently update VSAM files. RLS relies on the logging, commit coordination, and back out processing provided by CICS online systems. TVS adds its own logging, commit coordination, and back out support. This allows batch update jobs to share VSAM files between each other and CICS. PRA is using this same capability to allow multiple DBRC instances to do concurrent updates to the RECONs.

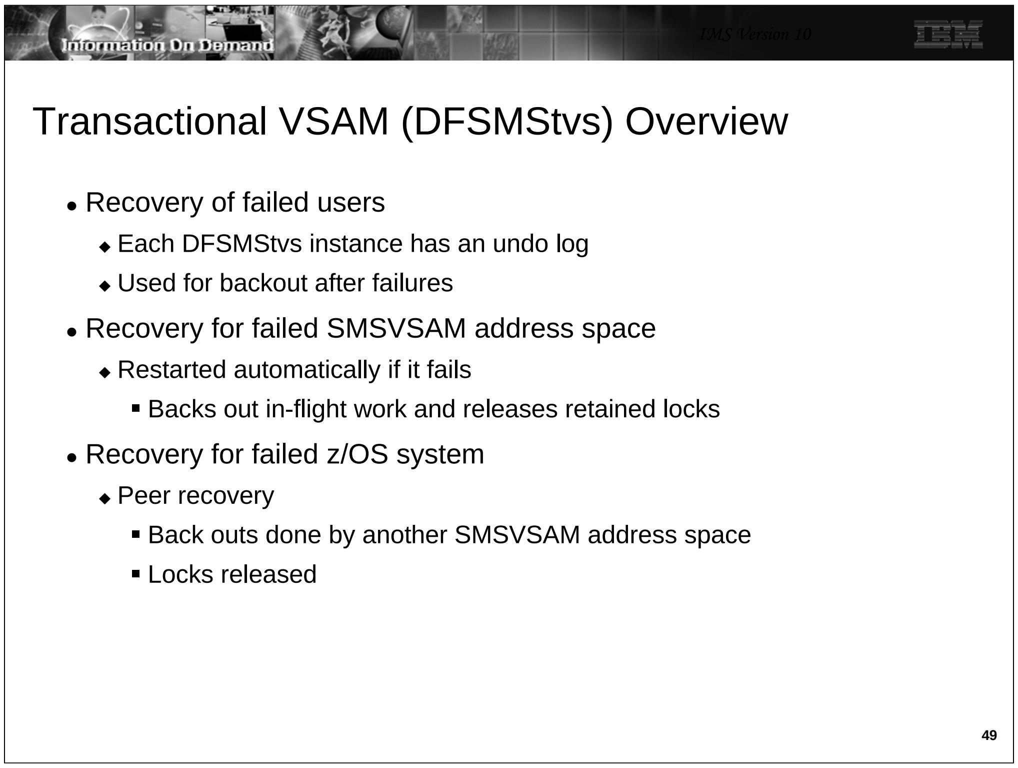
RLS and TVS execute in the SMSVSAM address space. This is a system address space that is typically started at z/OS initialization. There is only one SMSVSAM address space per LPAR. It provides RLS and TVS services to all users in the LPAR. This includes all instances of DBRC.

DBRC RECONs - DFSMStvs Environment



SMSVSAM Coupling Facility Structures
- Lock structure
- Cache structure
- Logging structures
- RRS logging structures

This is an illustration of a DFSMStvs environment with multiple instances of DBRC using PRA. There is an SMSVSAM address space in both z/OS systems. This address space provides TVS services to the DBRCs in the LPAR. The illustration shows that there is an IMS online system, an IMS batch job (DLI or DBB), and a IMS utility in each system. TVS uses the system logger and RRS. RRS also uses the system logger. The system logger has structures in the coupling facility. The SMSVSAM address space also connects to the coupling facility for its own structures. These are cache structures and a lock structure. There are a pair of share control data sets (SHCDS) for the VSAM data sets. These data sets contain information about the use of VSAM data sets that are being shared. There is also a spare SHCDS. Of course, there are the RECON pair and spare.



Transactional VSAM (DFSMStvs) Overview

- Recovery of failed users
  - Each DFSMStvs instance has an undo log
  - Used for backout after failures
- Recovery for failed SMSVSAM address space
  - Restarted automatically if it fails
    - Backs out in-flight work and releases retained locks
- Recovery for failed z/OS system
  - Peer recovery
    - Back outs done by another SMSVSAM address space
    - Locks released

49

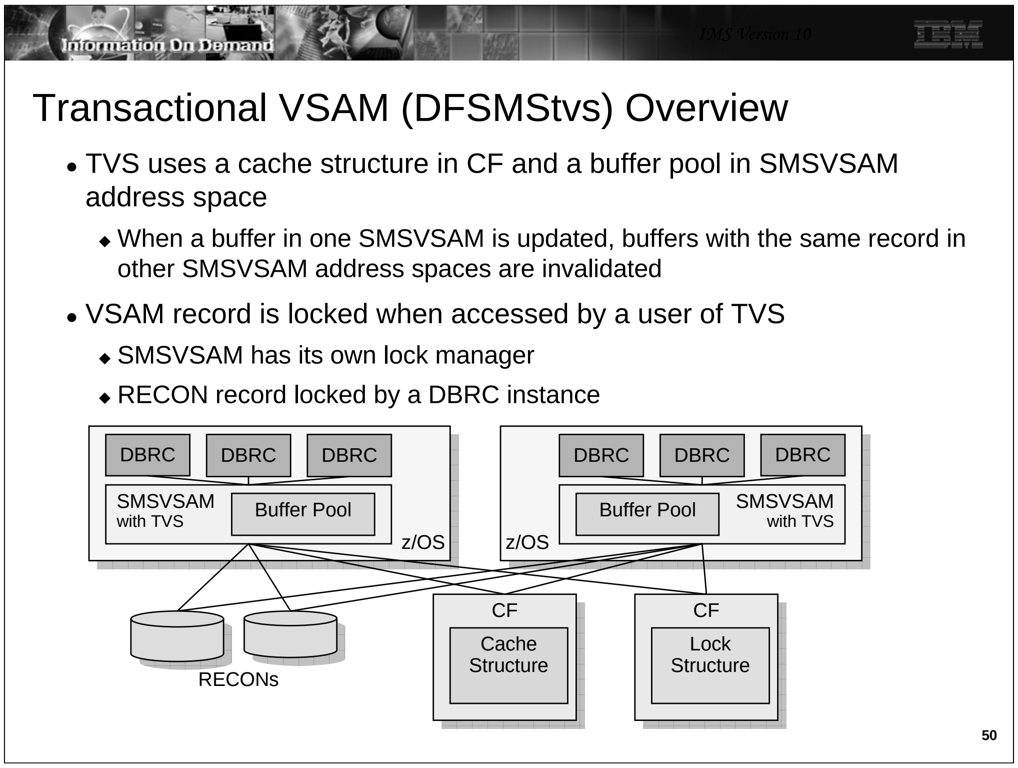
DFSMStvs has recovery capabilities to handle various kinds of failures.

Each DFSMStvs instance has its own undo log. These are log records which have the "before images" of records. If a user of DFSMStvs, such as DBRC fails, its uncommitted updates are backed out by DFSMStvs. The undo log is used for this purpose.

If DFSMStvs fails or the SMSVSAM address space fails, it is automatically restarted. When it restarts is backs out any in-flight work and releases any locks held by the in-flight transactions.

If a z/OS system fails, peer recovery is invoked. Back outs are done for the failed DFSMStvs by using another SMSVSAM address space in the Parallel Sysplex. The locks held for the failed work are released.

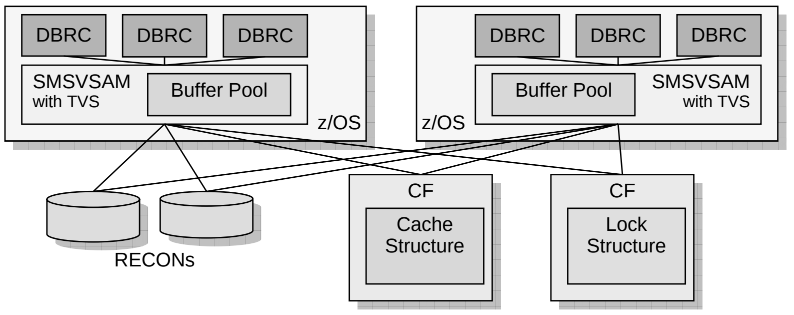
More information on these processes is provided later.



IMS Version 10

Transactional VSAM (DFSMStvs) Overview

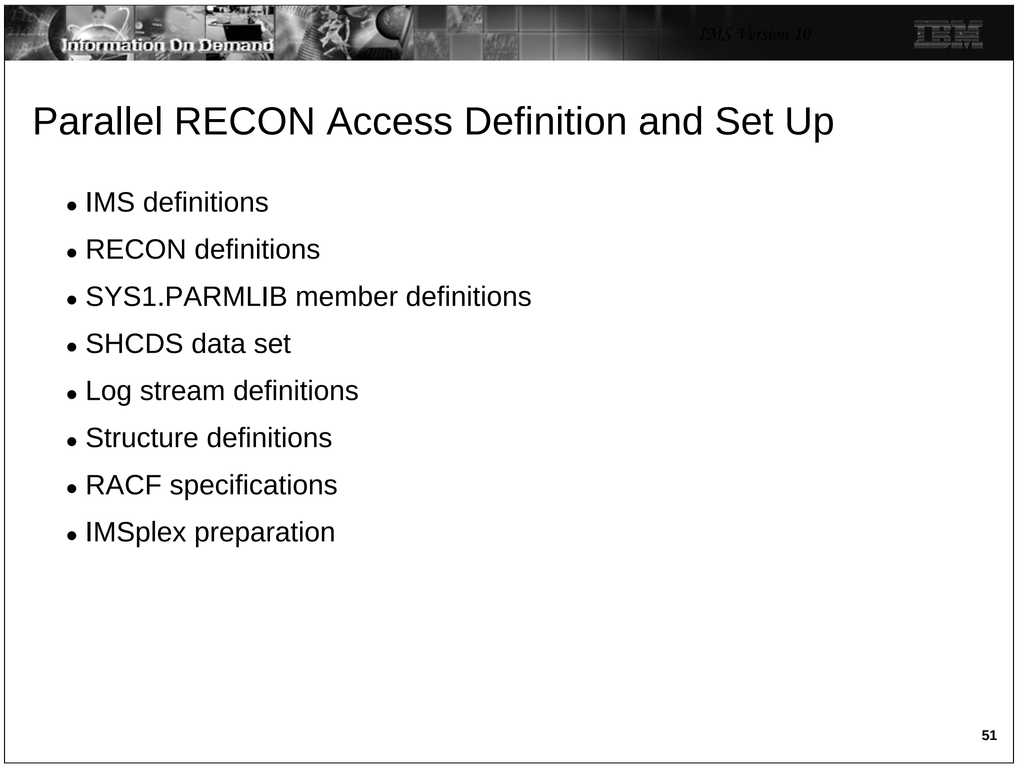
- TVS uses a cache structure in CF and a buffer pool in SMSVSAM address space
  - When a buffer in one SMSVSAM is updated, buffers with the same record in other SMSVSAM address spaces are invalidated
- VSAM record is locked when accessed by a user of TVS
  - SMSVSAM has its own lock manager
  - RECON record locked by a DBRC instance



50

This illustrates a DFSMStvs environment with DBRC users. When a DBRC instance accesses a VSAM record in the RECONs, the request is processed by TVS in the SMSVSAM address space. The SMSVSAM address space has its own buffers. The buffer pools are maintained in data spaces owned by the SMSVSAM address space. When using TVS, DBRC does not have its own buffers for RECON processing. Instead, it uses the SMSVSAM buffers. These buffers are shared with all users of the SMSVSAM address space. These are all of the VSAM data sets in the z/OS system which are using RLS or TVS. SMSVSAM also uses cache structures in the Coupling Facilities. Each shared data set is assigned to a structure. When a VSAM record in an SMSVSAM buffer pool is updated, if it resides in buffers in other systems those buffers are invalidated. This is done using the cross invalidation capability associated with cache structures and Parallel Sysplex.

The processing of VSAM requests includes lock requests. SMSVSAM has its own lock manager. It does not use the IRLM, however, its lock manager provides functions similar to those provided by IRLM for IMS and DB2 databases. The owner of a lock request is the DBRC instance. This is an IMS online system, an IMS batch job, or an IMS utility. Lock information is held in a lock structure in the CF.

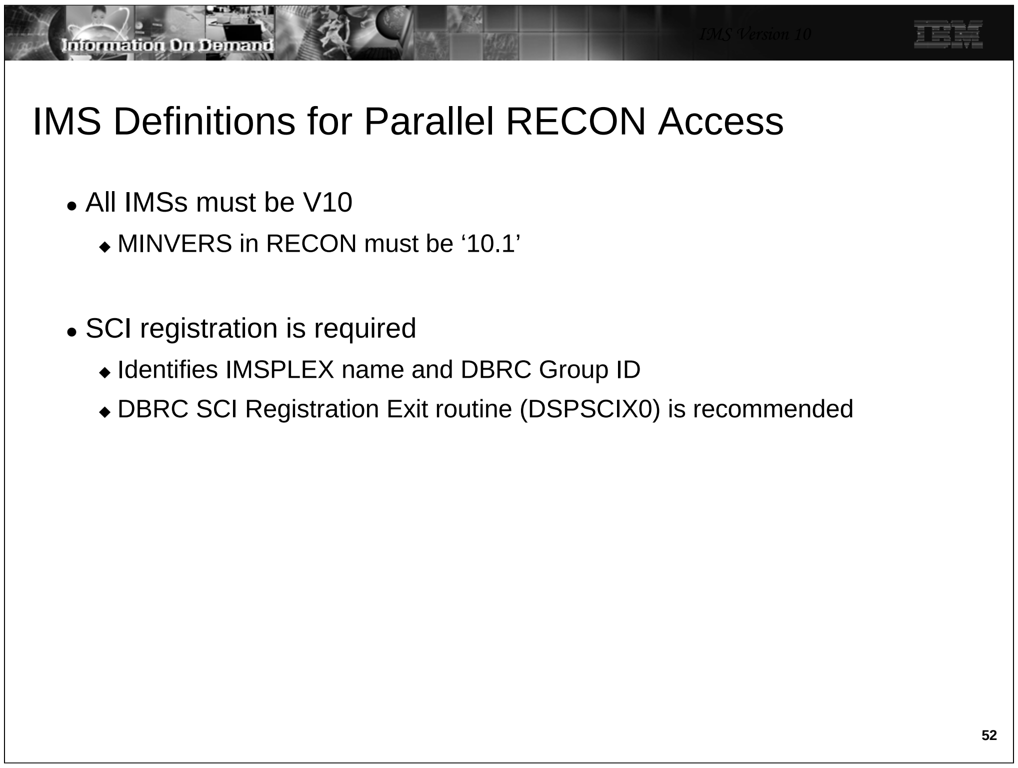


Parallel RECON Access Definition and Set Up

- IMS definitions
- RECON definitions
- SYS1.PARMLIB member definitions
- SHCDS data set
- Log stream definitions
- Structure definitions
- RACF specifications
- IMSplex preparation

51

Defining Parallel RECON Access includes several specifications. These will be described on the following pages.



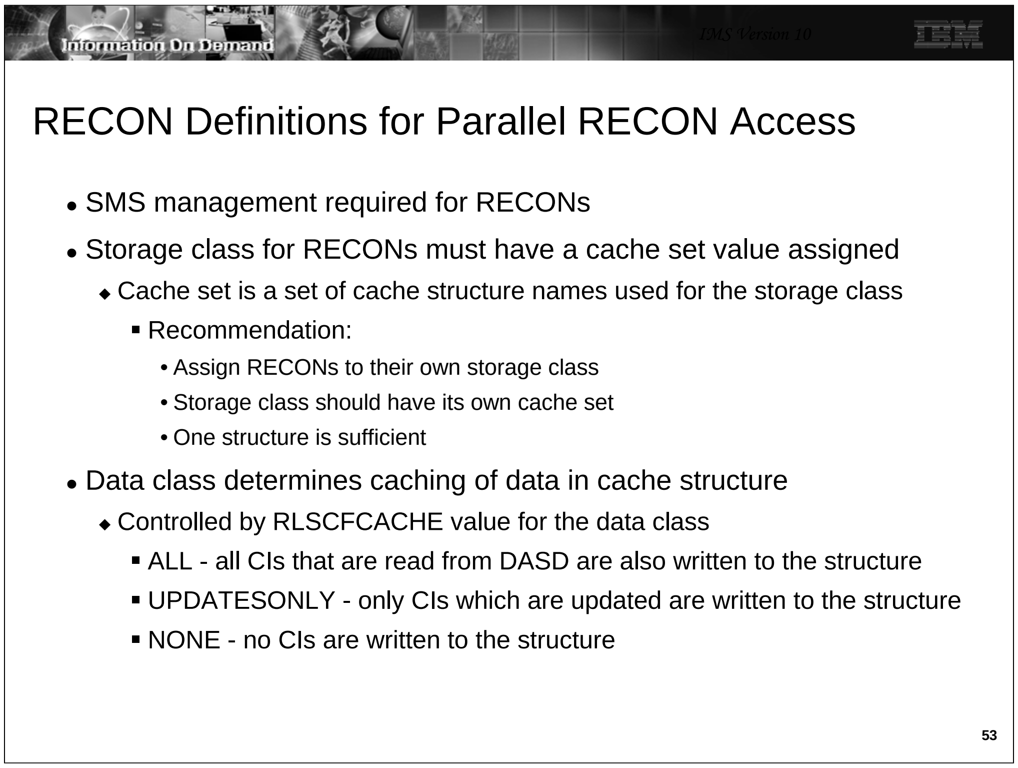
IMS Definitions for Parallel RECON Access

- All IMSs must be V10
  - MINVERS in RECON must be ‘10.1’
- SCI registration is required
  - Identifies IMSPLEX name and DBRC Group ID
  - DBRC SCI Registration Exit routine (DSPSCIX0) is recommended

52

PRA requires IMS V10. To ensure that all users of a set of RECONs are using V10, a MINVERS value of '10.1' must be specified for the RECONs. This prevents any lower release level of IMS from using the RECONs.

SCI registration is required for PRA. This means that a CSL environment is required. Each system where an IMS using PRA will be run must have an SCI address space. Since SCI registration is required, the use of a DBRC SCI Registration Exit routine (DSPSCIX0) is recommended. This ensures that consistent registration is done by all DBRC users and it eliminates the need to add the IMSPLEX= parameter to the JCL for online systems, batch jobs, and utilities. In addition to the IMSPLEX value which is assigned by the IMSPLEX= parameter of the exit routine, IMS V10 has added the DBRC group ID. This ID may also be assigned by the exit routine.



RECON Definitions for Parallel RECON Access

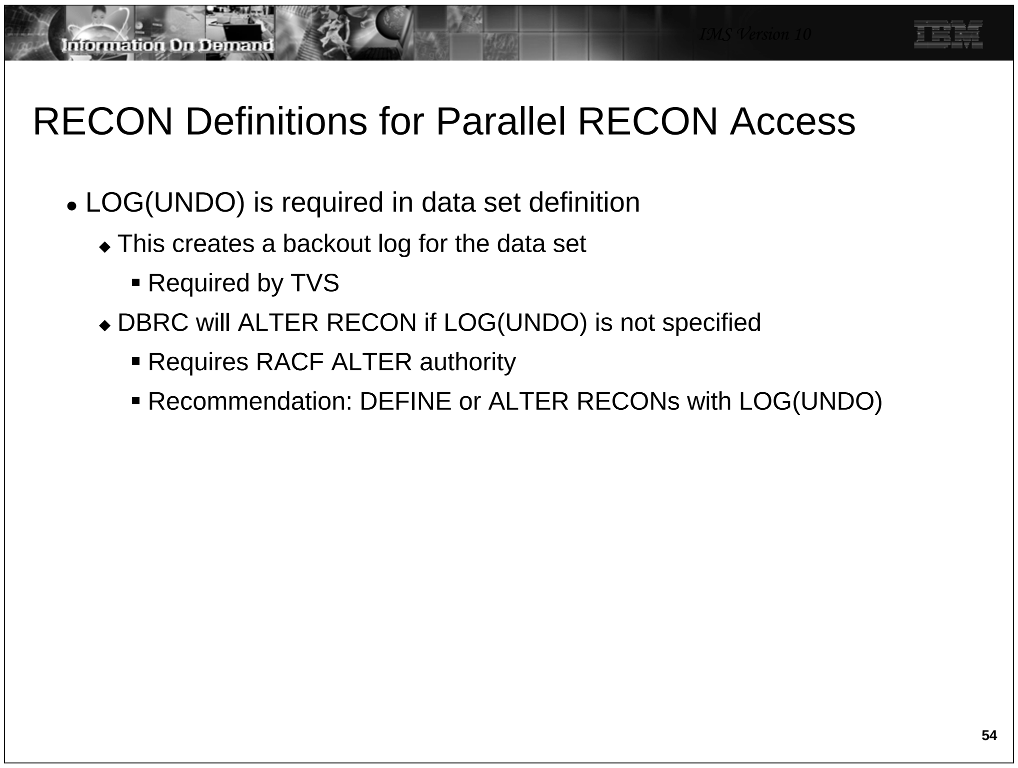
- SMS management required for RECONs
- Storage class for RECONs must have a cache set value assigned
  - Cache set is a set of cache structure names used for the storage class
    - Recommendation:
      - Assign RECONs to their own storage class
      - Storage class should have its own cache set
      - One structure is sufficient
- Data class determines caching of data in cache structure
  - Controlled by RLSCFCACHE value for the data class
    - ALL - all CIs that are read from DASD are also written to the structure
    - UPDATESONLY - only CIs which are updated are written to the structure
    - NONE - no CIs are written to the structure

53

When using PRA the RECONs must be under SMS management.

The RECONs must be defined with a storage class which has a cache set value. A cache set value assigns CF cache structures to the storage class. Many data sets (spheres) may have the same storage class. Multiple storage classes may use the same cache set. You can simplify the management and understanding of caching for the RECONs by assigning the RECON data sets to a storage class with it own cache set. The cache set needs only one structure defined to it. If the RECONs are the only RLS or TVS data sets in the storage class, they will have their own cache structure.

The data class for the RECONs determines the caching option. This is controlled by the RLSCFCACHE parameter for the data class. ALL specifies that all CIs read from DASD are also stored in the structure. UPDATESONLY specifies that CIs which are updated are written to the structure when they are written to DASD. NONE specifies that no CIs are stored in the structure. When NONE is specified, the structure is used only for buffer invalidation processing. There is another parameter which affects caching. This is the RLS_MAXCFFEATURELEVEL parameter on the IGDSMSxx member of SYS1.PARMLIB. This will be explained when IGDSMSxx is covered later in this section of the class.


Information On Demand
IMS Version 10

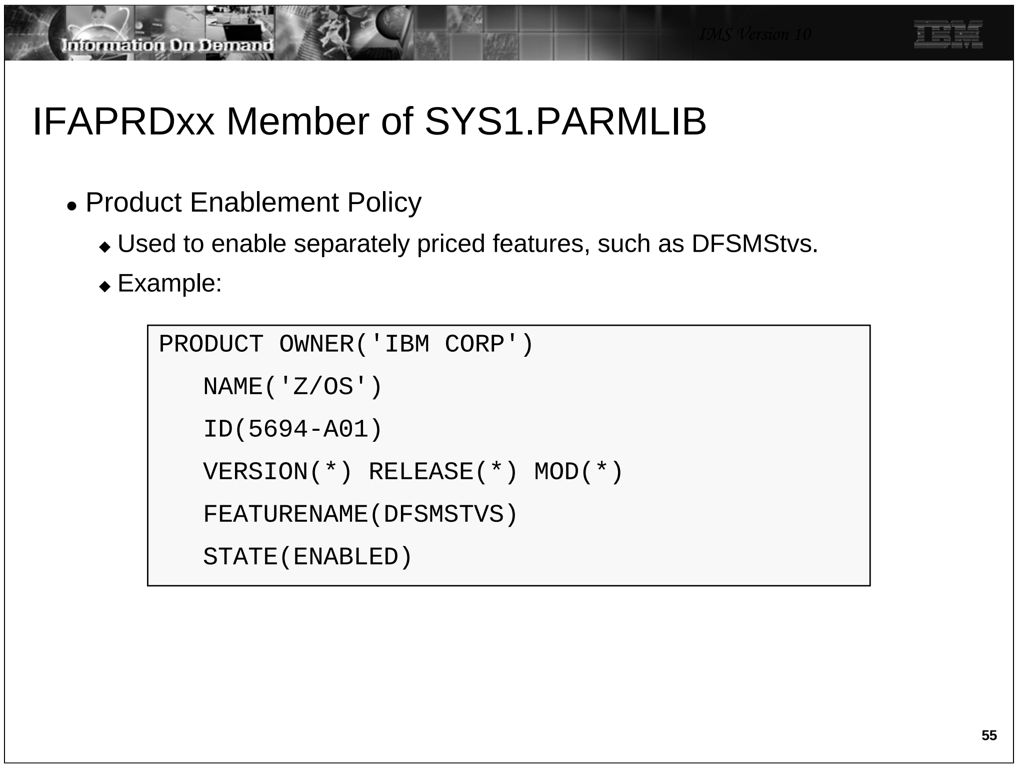
RECON Definitions for Parallel RECON Access

- LOG(UNDO) is required in data set definition
  - This creates a backout log for the data set
    - Required by TVS
  - DBRC will ALTER RECON if LOG(UNDO) is not specified
    - Requires RACF ALTER authority
    - Recommendation: DEFINE or ALTER RECONs with LOG(UNDO)

54

LOG(UNDO) is required in the definition of the RECONs for the use of Transactional VSAM. This creates a backout log for the data set when TVS is used. DBRC automatically sets this definition. If Parallel RECON Access is being used and LOG(UNDO) is not specified for the RECONs, DBRC will issue an ALTER to set the parameter to LOG(UNDO). If Parallel RECON Access is not being used, DBRC will ALTER RECON to LOG(NONE) if another value is defined. Of course, you can set LOG(UNDO) by issuing the DEFINE or ALTER with IDCAMS. In either case, the ALTER requires ALTER security from the security system, such as RACF.

Do not specify LOG(ALL) or a LOGSTREAMID parameter for the RECONs. LOG(ALL) creates a forward recovery log which is identified by the LOGSTREAMID. DBRC does not support forward recovery of the RECONs.



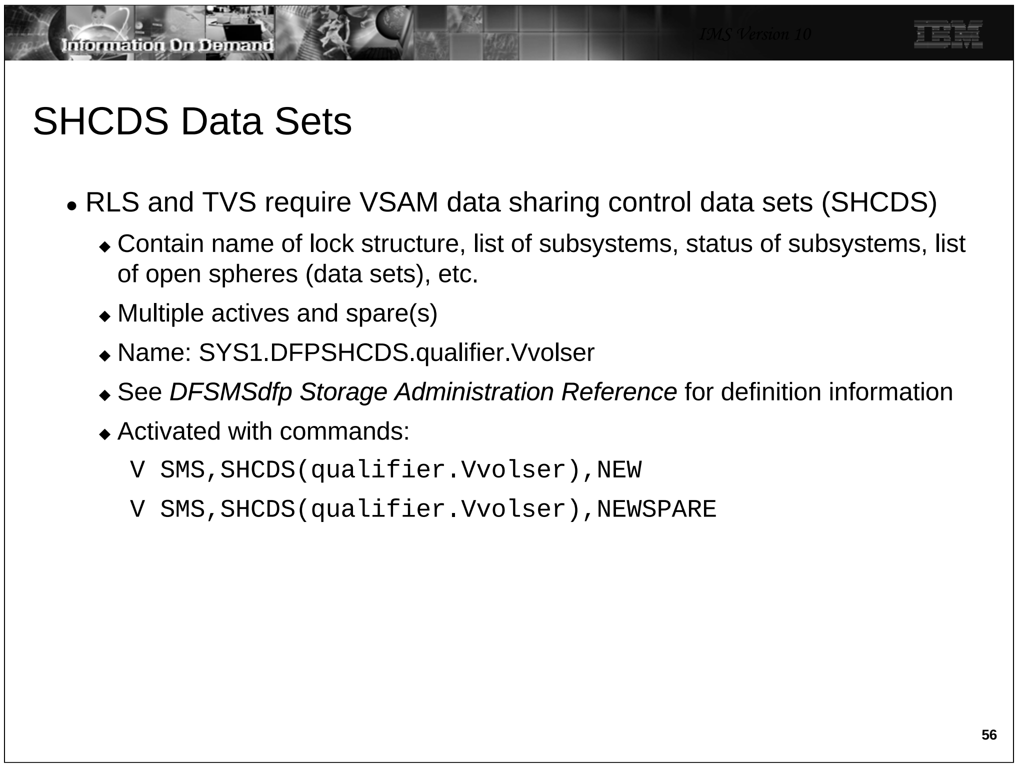
IFAPRDxx Member of SYS1.PARMLIB

- Product Enablement Policy
  - Used to enable separately priced features, such as DFSMStvs.
  - Example:

```
PRODUCT OWNER('IBM CORP')
   NAME('Z/OS')
   ID(5694-A01)
   VERSION(*) RELEASE(*) MOD(*)
   FEATURENAME(DFSMSTVS)
   STATE(ENABLED)
```

55

DFSMStvs is a separately priced feature of z/OS. It must be enabled with a product enablement policy which is defined in the IFAPRDxx member of SYS1.PARMLIB.



SHCDS Data Sets

- RLS and TVS require VSAM data sharing control data sets (SHCDS)
  - Contain name of lock structure, list of subsystems, status of subsystems, list of open spheres (data sets), etc.
  - Multiple actives and spare(s)
  - Name: SYS1.DFPSHCDS.qualifier.Vvolser
  - See *DFSMSdfp Storage Administration Reference* for definition information
  - Activated with commands:

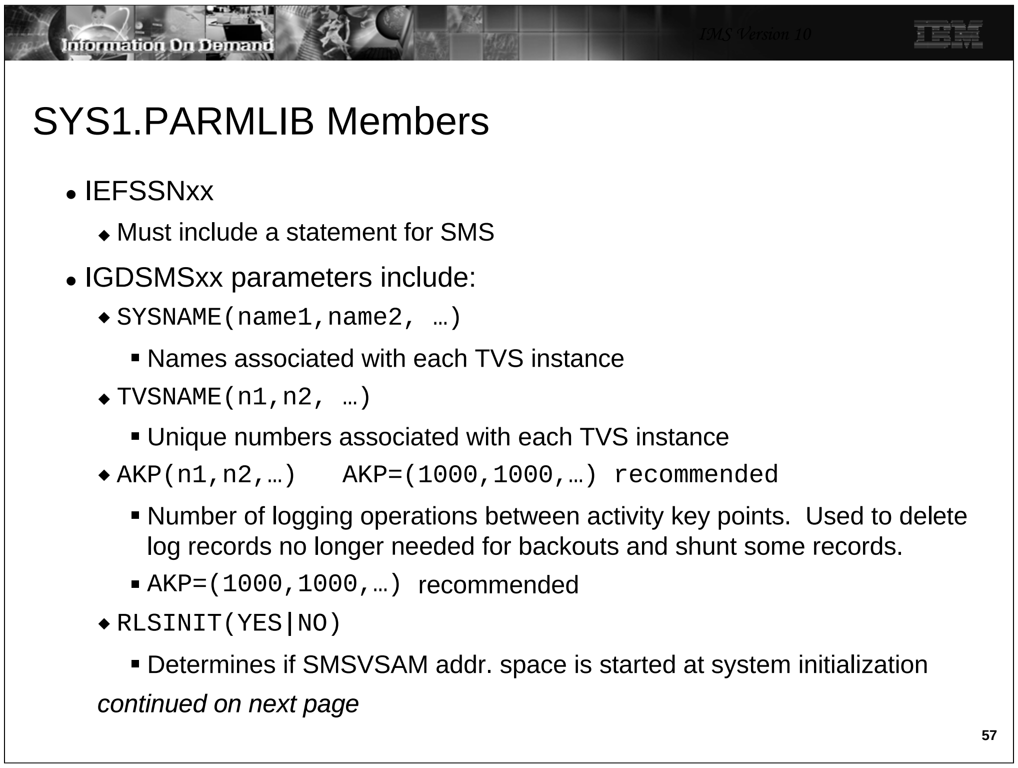
```
V SMS,SHCDS(qualifier.Vvolser),NEW
V SMS,SHCDS(qualifier.Vvolser),NEWSPARE
```

If you have not implemented RLS, you must defined SHCDS data sets. If you already have implemented RLS, you already have these data sets defined. These data sets are used to hold the name of the lock structure, the list of subsystems, and a list of open VSAM spheres (data sets). It also contains information about the spheres such as which subsystems have them open.

There can be multiple SHCDS data sets. Two is typical. Spares may also be defined. Having a pair and a spare provides recovery capabilities similar to those for a pair and spare for the RECONs.

The data set name for an SHCDS is always of the form SYS1.DFPSHCDS.qualifier.Vvolser where qualifier is whatever you want it to be and volser is the volume serial for its volume. See the *DFSMSdfp Storage Administration Reference* manual for complete information on defining SHCDS data sets.

VARY commands are used to make a data set an active SHCDS or a spare.


Information On Demand
IMS Version 10

SYS1.PARMLIB Members

- IEFSSNxx
  - Must include a statement for SMS
- IGDSMSxx parameters include:
  - `SYSNAME(name1,name2, …)`
    - Names associated with each TVS instance
  - `TVSNAME(n1,n2, …)`
    - Unique numbers associated with each TVS instance
  - `AKP(n1,n2,…)   AKP=(1000,1000,…) recommended`
    - Number of logging operations between activity key points. Used to delete log records no longer needed for backouts and shunt some records.
    - `AKP=(1000,1000,…)` recommended
  - `RLSINIT(YES|NO)`
    - Determines if SMSVSAM addr. space is started at system initialization

continued on next page

57

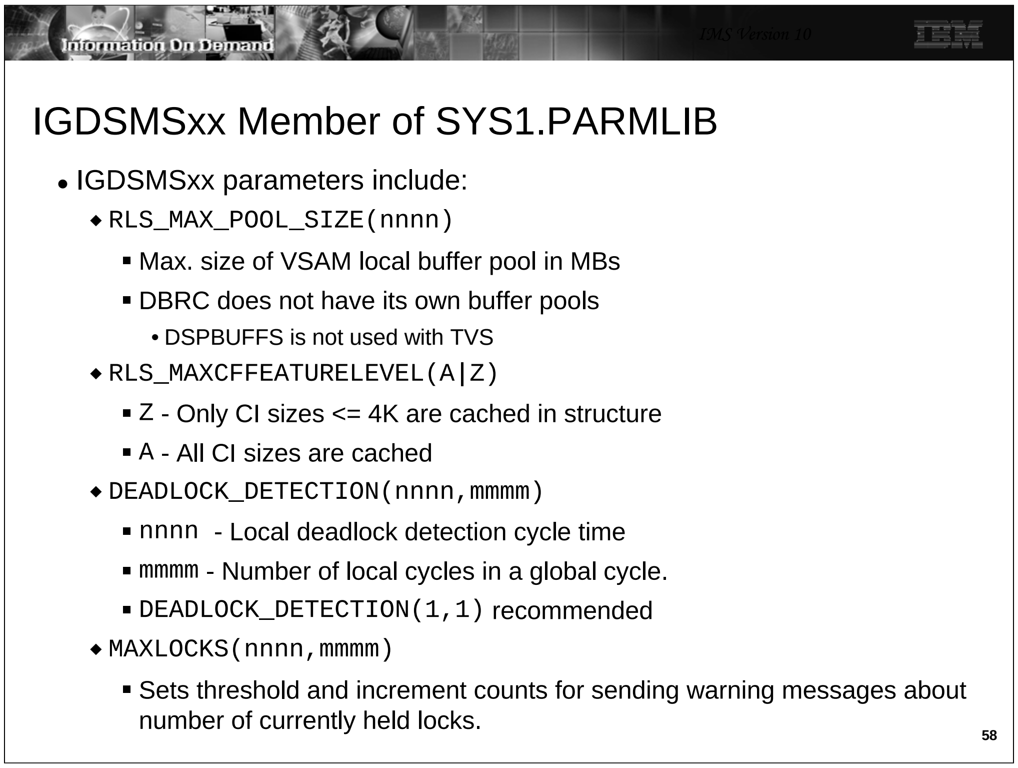
SMS must be defined to MVS as a subsystem. This is done by including a statement for SMS in the IEFSSNxx SYS1.PARMLIB member.

The names associated with each TVS instance are defined in the SYSNAME parameter. Similarly, each TVS instance is assigned a unique number with the TVSNAME parameter.

AKP controls the occurrence of activity key points. Separate numbers may be specified for the different systems. At these times log records which are no longer needed for backouts are deleted. Log records for units of recovery that have not logged in two AKPs are moved to the shunt log.

The IGDSMSxx member of SYS1.PARMLIB is described in the *z/OS MVS Initialization and Tuning Guide*. Some of the most important parameters for TVS users are shown here.

If RLSINIT(YES) is not specified, the SMSVSAM address space must be started with a V SMSVSAM,ACTIVE command.



IGDSMSxx Member of SYS1.PARMLIB

- IGDSMSxx parameters include:
  - `RLS_MAX_POOL_SIZE(nnnn)`
    - Max. size of VSAM local buffer pool in MBs
    - DBRC does not have its own buffer pools
      - DSPBUFFS is not used with TVS
  - `RLS_MAXCFFEATURELEVEL(A|Z)`
    - `Z` - Only CI sizes <= 4K are cached in structure
    - `A` - All CI sizes are cached
  - `DEADLOCK_DETECTION(nnnn,mmmm)`
    - `nnnn` - Local deadlock detection cycle time
    - `mmmm` - Number of local cycles in a global cycle.
    - `DEADLOCK_DETECTION(1,1)` recommended
  - `MAXLOCKS(nnnn,mmmm)`
    - Sets threshold and increment counts for sending warning messages about number of currently held locks.

58

RLS_MAX_POOL_SIZE specifies in megabytes the maximum size of the VSAM local buffer pool in SMSVSAM. This pool is created in a data space. For serial access (not Parallel RECON Access) to the RECONs IMS uses the DSPBUFFS specification to determine the number of buffers used by a DBRC instance. This defaults to 60 index buffers and 120 data buffers. You may have specified a different value. As a starting point, you may make the SMSVSAM local buffer pool the size of the buffer pools for all of the concurrent DBRC instances that will be run. For example, if you took the default DSPBUFFS, have a RECON CI size of 16K, and have 10 concurrent DBRC instances, you could start with a buffer pool of 10 x (120 + 60) x 16K = 29M. Of course, a larger size might provide better performance by keeping more CIs in the pool and potentially avoiding some reads.

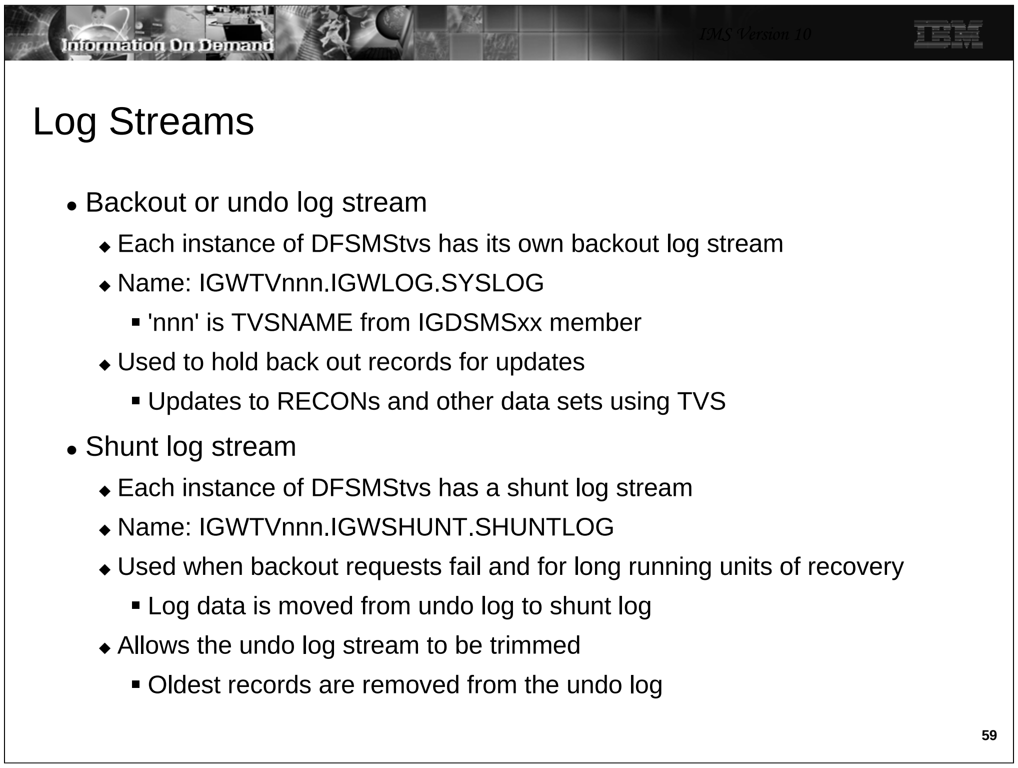
The RLS_MAXCFFEATURELEVEL parameter is used to specify if CIs larger than 4K will be cached in the SMSVSAM cache structure. 'Z' specifies that only 4K and smaller CIs will be cached. 'A' specifies that all may be cached. Since RECON CI sizes should be larger than 4K, you will need to specify 'A' to cache RECON CIs.

DEADLOCK_DETECTION(nnnn,nnnn) is used to specify the number of seconds in a local deadlock detection cycle and the number of local cycles in a global cycle. These default to 15 seconds and 4. Most installation will want faster deadlock detections. DEADLOCK_DETECTION(1,1) is recommended.

MAXLOCKS is used to control the issuing of warning messages about the number of locks held. These are messages IGW859I and IGW10074I. You might want to set these values to 50% and 80% of the number of record list entries in the lock structure.

RLSTMOUT may be specified but DBRC overrides this value. This is the time that a lock request will wait before it is timed out. DBRC always uses a value of 2 seconds.

CF_TIME is used to specify at what interval SMF type 42 records containing information and statistics on CF cache and lock structures are gathered.



Log Streams

- Backout or undo log stream
  - Each instance of DFSMStvs has its own backout log stream
  - Name: IGWTVnnn.IGWLOG.SYSLOG
    - 'nnn' is TVSNAME from IGDSMSxx member
  - Used to hold back out records for updates
    - Updates to RECONs and other data sets using TVS
- Shunt log stream
  - Each instance of DFSMStvs has a shunt log stream
  - Name: IGWTVnnn.IGWSHUNT.SHUNTLOG
  - Used when backout requests fail and for long running units of recovery
    - Log data is moved from undo log to shunt log
  - Allows the undo log stream to be trimmed
    - Oldest records are removed from the undo log

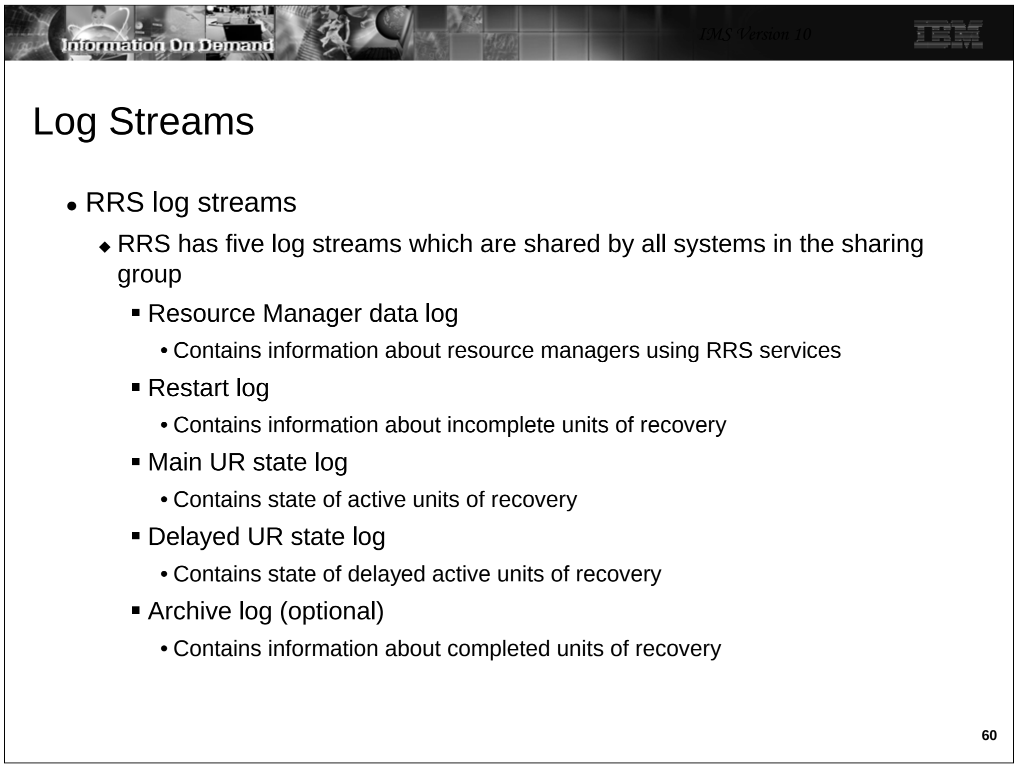
59

DFSMStvs maintains an "undo" log stream for each instance of SMSVSAM. This means that there is one per z/OS system using DFSMStvs. The log stream contains "before" images of updates made by users of this DFSMStvs instance. It is used to back out any updates that have been done by units of recovery which fail or are victims in a deadlock. The log stream name is IGWTVnnn.IGWLOG.SYSLOB where "nnn" is the TVSNAME assigned in the IGDSMSxx member of SYS1.PROCLIB.

Typically, backout records do not need to be maintained for a long time. When the unit of recovery is committed, these log records may be deleted. On the other hand, there are occasions when the log records need to be kept for a long time. This would occur if a backout failed due to the inability to do the backout updates to the data set or if a unit of recovery was not committed for a long time. In these cases, log records may be moved to a shunt log stream. Each instance of DFSMStvs has a shunt log stream with the name IGWTVnnn.IGWSHUNT.SHUNTLOG where "nnn" is the TVSNAME.

Log records are moved from the undo log to the shunt log by activity keypoint processing. The frequency of this processing is determined by the AKP parameter in the IGDSMSxx member. If a unit of recovery has not logged for two AKPs, its log records are moved to the shunt log.

The use of the shunt log allows TVS to trim the undo log. TVS tracks the oldest log entry for any running unit of recovery. It can trim records that are older than this. This limits the size of the undo log and tends to keep all of its records in the log structure.



Log Streams

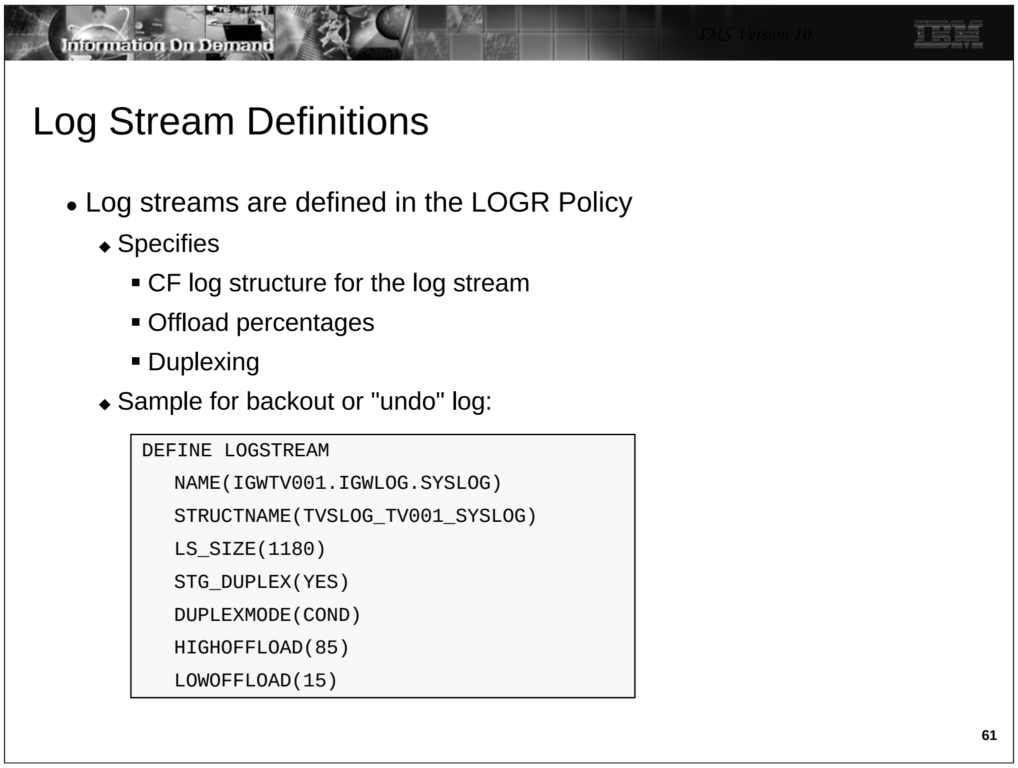
- RRS log streams
  - RRS has five log streams which are shared by all systems in the sharing group
    - Resource Manager data log
      - Contains information about resource managers using RRS services
    - Restart log
      - Contains information about incomplete units of recovery
    - Main UR state log
      - Contains state of active units of recovery
    - Delayed UR state log
      - Contains state of delayed active units of recovery
    - Archive log (optional)
      - Contains information about completed units of recovery

60

Each request to DBRC is a unit of recovery which is tracked and logged by RRS.

RRS has multiple log streams. These log streams are used for various purposes. DFSMStvs is only one of several components which use RRS. Other IMS users of RRS include APPC, ODBA, and OTMA. Most installations will already have RRS log streams defined.

Many installations have already implemented RRS. For example, IMS requires RRS for ODBA and APPC protected conversations which use SYNCLVL=SYNPT.



Log Stream Definitions

- Log streams are defined in the LOGR Policy
  - Specifies
    - CF log structure for the log stream
    - Offload percentages
    - Duplexing
  - Sample for backout or "undo" log:

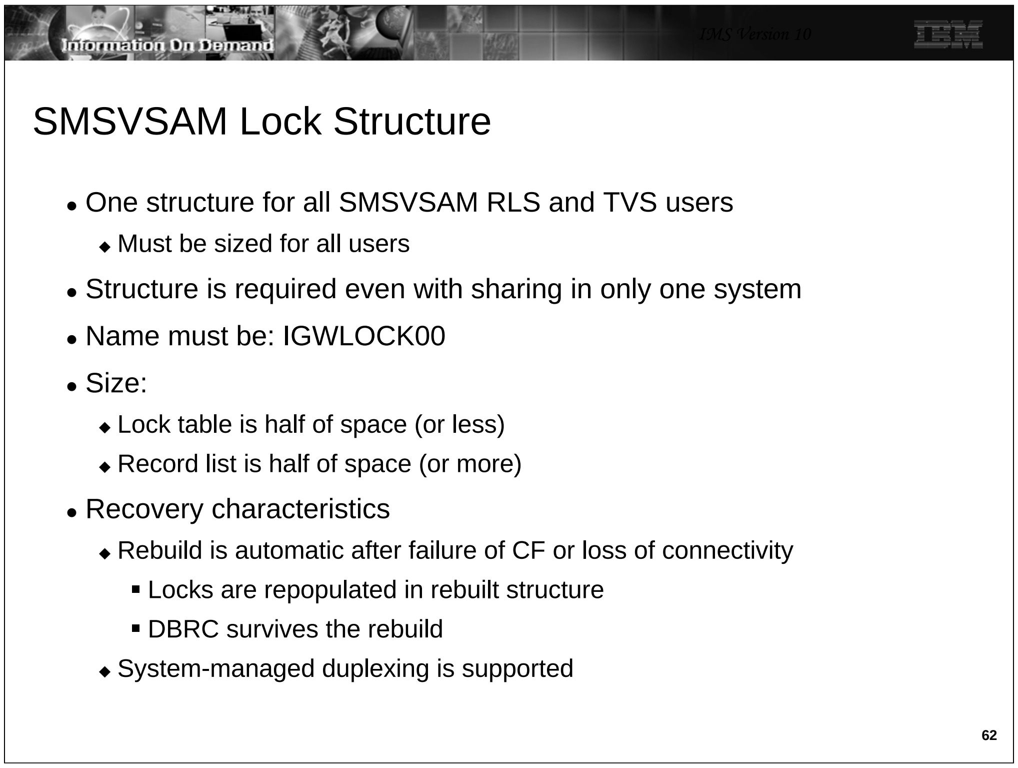
```
DEFINE LOGSTREAM
   NAME(IGWTV001.IGWLOG.SYSLOG)
   STRUCTNAME(TVSLOG_TV001_SYSLOG)
   LS_SIZE(1180)
   STG_DUPLEX(YES)
   DUPLEXMODE(COND)
   HIGHOFFLOAD(85)
   LOWOFFLOAD(15)
```

61

Log streams must be defined in the LOGR policy for a sysplex. Each log stream is defined with a DEFINE LOGSTREAM statement. The policy assigns the log stream to a structure. The CFRM policy defines the size of the structure.

The sample definition shows some of the other parameters that you may define. The LS_SIZE parameter defines size of the data used for offloading the log stream from the structure. The size is specified in 4k blocks. STG_DUPLEX(YES) specifies that the log stream is duplexed in staging data sets when the condition defined in the DUPLEXMODE parameter is satisfied. DUPLEXMODE(COND) specifies that duplexing occurs when the CF structure is in a volatile CF or when the CF and the SMSVSAM address space reside in the same machine (CPC). The HIGHOFFLOAD parameter specifies the percentage of the structure that must be full before an offload process occurs. Offload moves log records from the structure to an offload data set. LOWOFFLOAD specifies the percentage of the log stream that may remain in the structure when an offload process completes. All parameters that you may specify in a LOGR policy are documented in the *z/OS MVS Setting Up a Sysplex* manual.

Since DBRC "transactions" that update the RECONs should be short-lived, undo logs for DBRC should be small. They are deleted by activity keypoint processing (AKP). In addition, the log records for long-lived transactions will be moved to the shunt log by activity keypoint processing (AKP).



SMSVSAM Lock Structure

- One structure for all SMSVSAM RLS and TVS users
  - Must be sized for all users
- Structure is required even with sharing in only one system
- Name must be: IGWLOCK00
- Size:
  - Lock table is half of space (or less)
  - Record list is half of space (or more)
- Recovery characteristics
  - Rebuild is automatic after failure of CF or loss of connectivity
    - Locks are repopulated in rebuilt structure
    - DBRC survives the rebuild
  - System-managed duplexing is supported

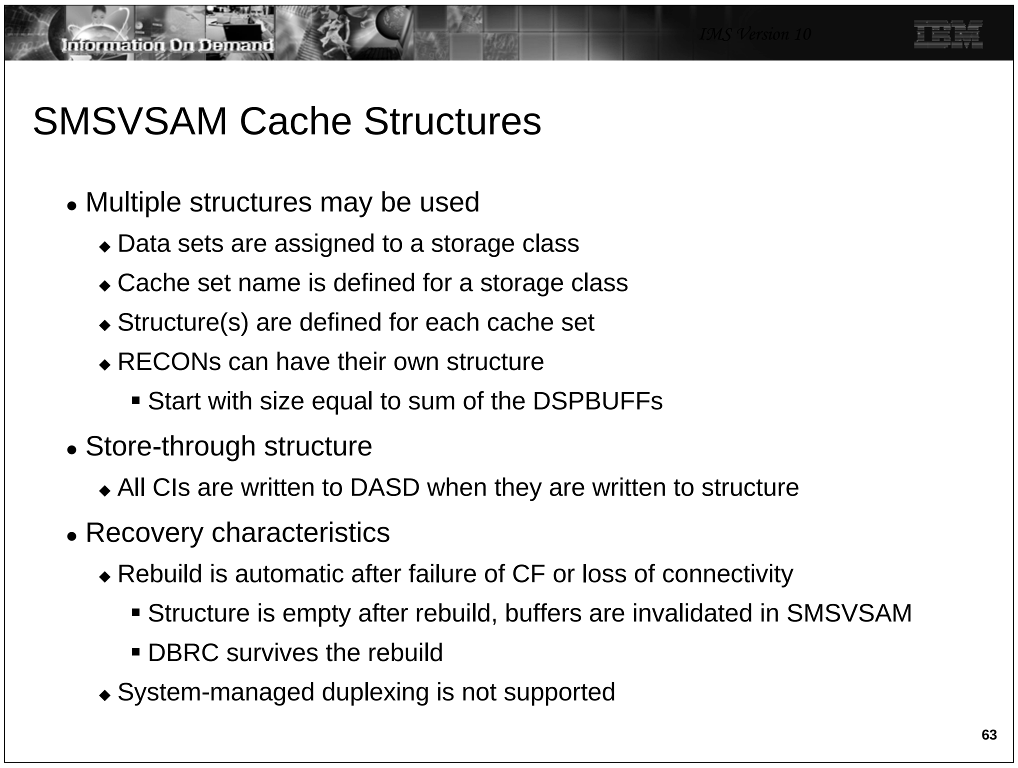
62

SMSVSAM uses a lock structure. It is always required, even when there is only one system using RLS or TVS. The lock structure holds locks for all of the data sets using RLS and TVS. This may include data sets other than RECONs. It must be sized to hold all of these locks.

The lock structure name is always IGWLOCK00. If the size of the lock structure is a power of 2, it is equally divided between the lock table and the record list. If it is not a power of two, the size of the lock table is the largest power of 2 that is smaller than half the structure size. The remaining space is used for the record list. Most DBRC actions do not hold many locks. These actions also typically hold locks for a short time. There is one exception. A LIST of the RECONs with the STATIC option gets a lock on every record. These locks are held until the LIST completes. A LIST with the STATIC option is generally not recommended for this reason.

The lock structure is automatically rebuilt after a failure of the lock structure, a failure of its CF, or loss of connectivity to its CF. If the SMSVSAM address spaces survive, the locks are repopulated in the rebuilt structure. DBRC will survive this recovery. This applies to all instances of DBRC including online systems, IMS batch jobs, and IMS utilities using DBRC.

System-managed duplexing may be used for the lock structure. Duplexing is not recommended when the structure is placed on a machine that does not contain an SMSVSAM address space connected to the structure. The overhead of duplexing is unlikely to be justified in this case since recovery is done without the loss of any users.



SMSVSAM Cache Structures

- Multiple structures may be used
  - Data sets are assigned to a storage class
  - Cache set name is defined for a storage class
  - Structure(s) are defined for each cache set
  - RECONs can have their own structure
    - Start with size equal to sum of the DSPBUFFs
- Store-through structure
  - All CIs are written to DASD when they are written to structure
- Recovery characteristics
  - Rebuild is automatic after failure of CF or loss of connectivity
    - Structure is empty after rebuild, buffers are invalidated in SMSVSAM
    - DBRC survives the rebuild
  - System-managed duplexing is not supported

63

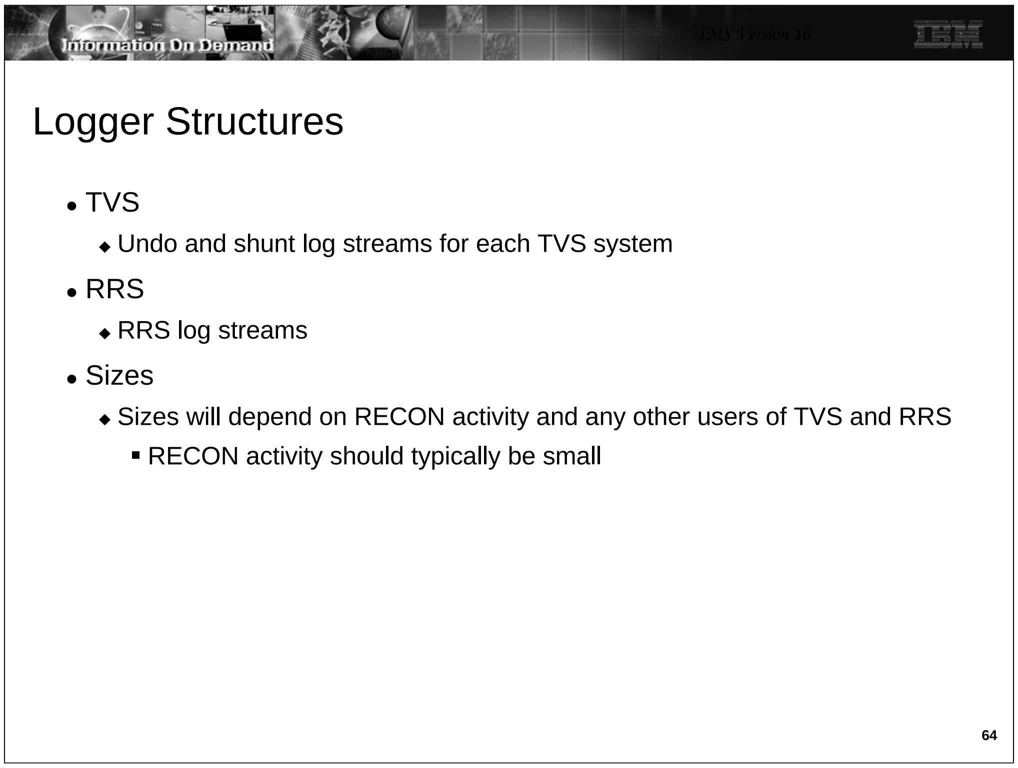
SMSVSAM uses one or multiple cache structures. Data sets are assigned to structures through their storage classes. A set of structures may be assigned to a storage class. The data sets in the storage class use the structures defined for the storage class. This allows you to set aside a structure for the exclusive use of the RECONs. By doing this you may size a structure to meet the needs of the RECONs. As a starting point you could make the structure large enough to hold all of the data that was previously in the DBRC buffers. These are defined with DSPBUFFS.

When storage class has multiple cache structures, data sets are assigned to a structure when they are opened. SMSVSAM attempts to balance the use of structures in a storage class.

The SMSVSAM cache structures are store-through structures. This means that all of the CIs that are written to the structure are also written to the RECON data sets. All committed updates are in the data sets.

If a Coupling Facility containing an SMSVSAM cache structure fails, the structure is automatically rebuilt on another CF. When the structure is rebuilt, it is not repopulated. All of the buffers containing CIs assigned to the structure are invalidated. This is similar to the actions that IMS takes when a full function cache structure is lost. If a structure cannot be rebuilt but the storage class has one or more other structures, the data sets using the failed structure are reassigned to another structure. DBRC survives the rebuilding of structures and the reassignment of data sets to other structures. These failures and recoveries are masked from DBRC by SMSVSAM.

System-managed duplexing is not supported for SMSVSAM cache structures. It is not needed since the loss of a structure is easily handled by the rebuild done by SMSVSAM.


Information On Demand
IMS Version 10

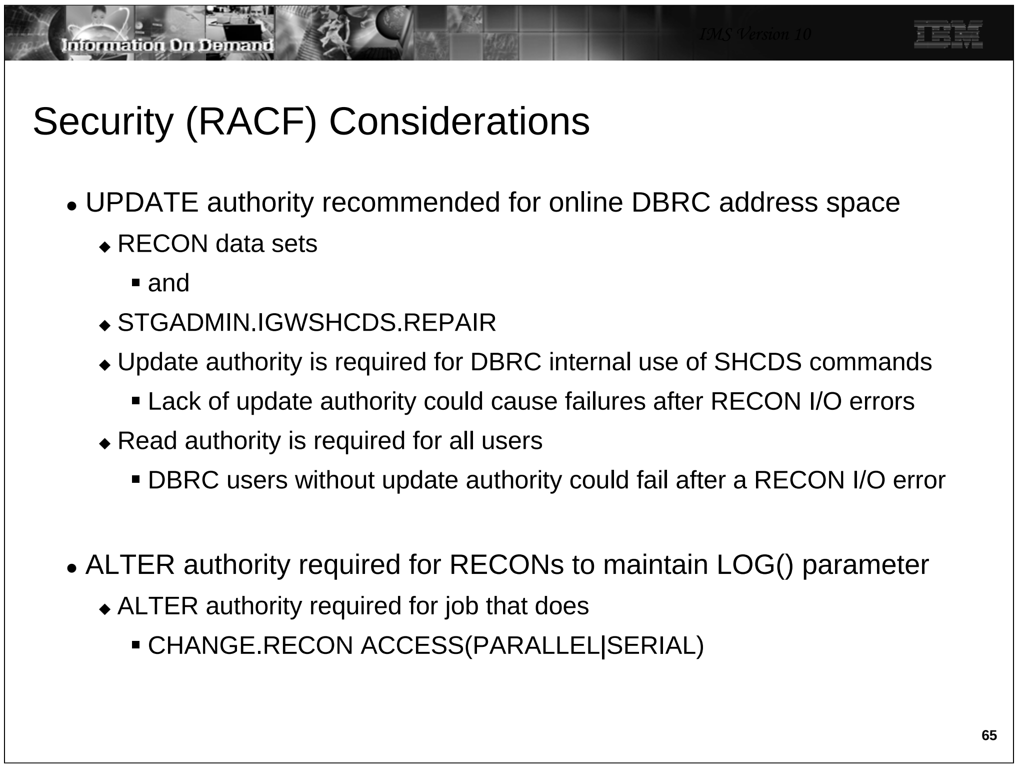
Logger Structures

- TVS
  - Undo and shunt log streams for each TVS system
- RRS
  - RRS log streams
- Sizes
  - Sizes will depend on RECON activity and any other users of TVS and RRS
    - RECON activity should typically be small

64

Log structures are required for the DFSMStvs undo and shunt logs and for the RRS log stream.

PRA will typically not require much space in these structures. PRA "transactions" typically make few updates to the RECONs and do not wait for non-DBRC work. This tends to make its space requirements small. If a structure is not large enough, records are offloaded to logger data sets. This offloading depends on the size of the structure, the sizes of the log streams using the structure, and the HIGHOFFLOAD and LOWOFFLOAD parameters in the log stream definition in the LOGR policy.



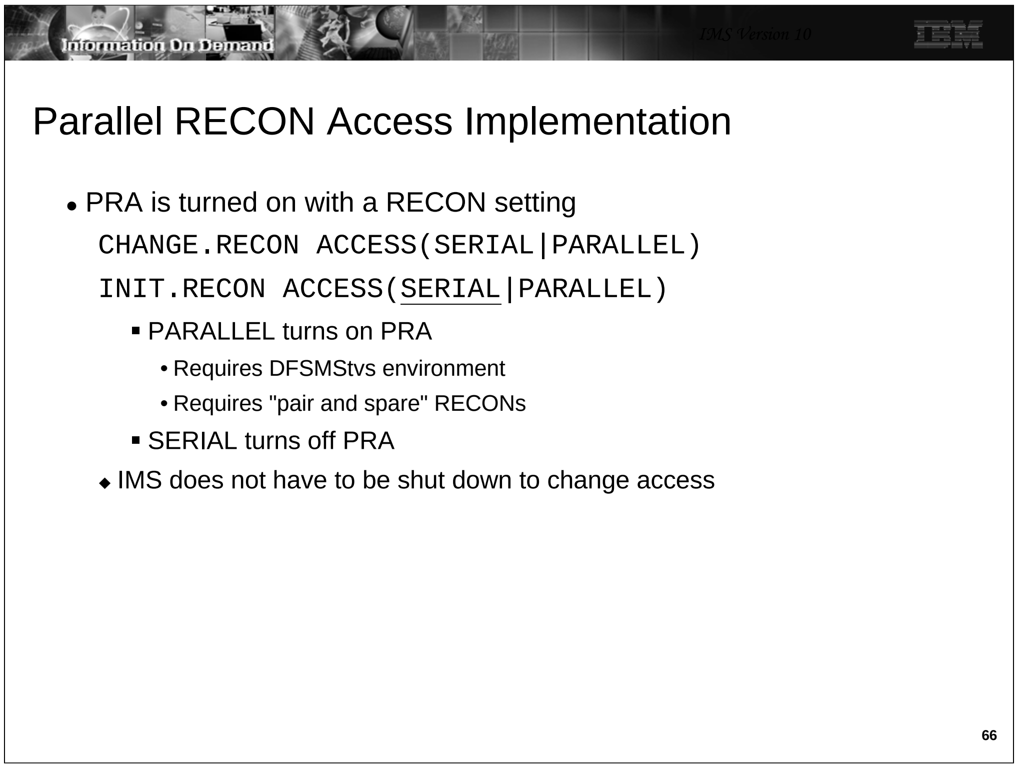
Security (RACF) Considerations

- UPDATE authority recommended for online DBRC address space
  - RECON data sets
    - and
  - STGADMIN.IGWSHCDS.REPAIR
  - Update authority is required for DBRC internal use of SHCDS commands
    - Lack of update authority could cause failures after RECON I/O errors
  - Read authority is required for all users
    - DBRC users without update authority could fail after a RECON I/O error

- ALTER authority required for RECONs to maintain LOG() parameter
  - ALTER authority required for job that does
    - CHANGE.RECON ACCESS(PARALLEL|SERIAL)

DBRC issues some DFSMS commands for recovery situations. These commands require update authority to the RECONs and the STGADMIN.IGWSHCDS.REPAIR resource. Read authority is required for all users for the RECONs and the "REPAIR" resource. Update authority is required in some recovery situations. If a RECON I/O error occurs, it is likely that update authority will be required for the recovery to the spare RECON. A lack of update authority would result in the user failing. Obviously, it would be good for no users to fail, however, it is probably essential that online systems not fail.

DBRC ALTERs the RECONs when switching from serial to parallel access or from parallel to serial access. This should not be a typical operation. The ALTER is used to change from LOG(NONE) to LOG(UNDO) or from LOG(UNDO) to LOG(NONE). ALTER authority is required for the DBRC jobs that issue the command. DBRC also issues an ALTER when reconfiguring the RECONs if the spare has the wrong setting for the LOG parameter. If the spare has been defined with the correct LOG value, the ALTER is not required.



Parallel RECON Access Implementation

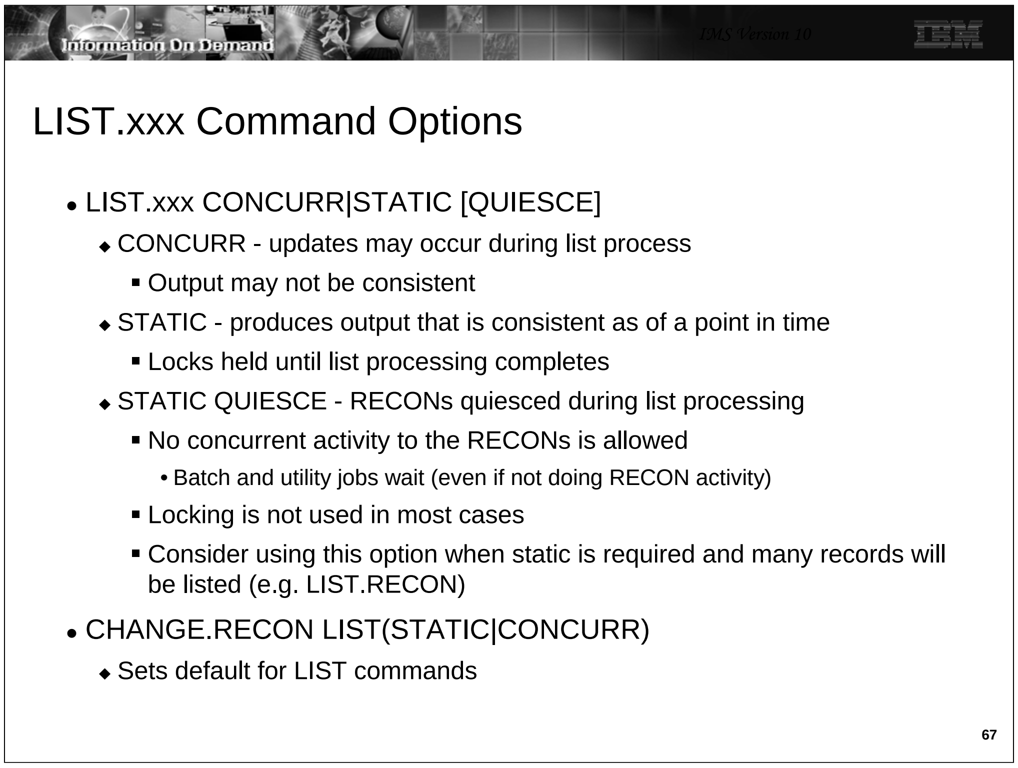
- PRA is turned on with a RECON setting
  `CHANGE.RECON ACCESS(SERIAL|PARALLEL)`
  `INIT.RECON ACCESS(SERIAL|PARALLEL)`
  - PARALLEL turns on PRA
    - Requires DFSMStvs environment
    - Requires "pair and spare" RECONs
  - SERIAL turns off PRA
  - IMS does not have to be shut down to change access

Parallel RECON Access is turned on by specifying ACCESS(PARALLEL) on the CHANGE.RECON or INIT.RECON command. It may be turned off with ACCESS(SERIAL). SERIAL is the default for the INIT.RECON command.

Since PRA requires DFSMStvs a complete DFSMStvs environment must be active when the command is issued. For example, the command must be issued on a z/OS system with an SMSVSAM address space and the lock and cache structures must be defined.

A good pair of active RECONs and a spare RECON must be available when the CHANGE.RECON ACCESS(PARALLEL) command is issued.

There cannot be an active RSR tracking subsystem when a change to parallel access is made.



LIST.xxx Command Options

- LIST.xxx CONCURR|STATIC [QUIESCE]
  - CONCURR - updates may occur during list process
    - Output may not be consistent
  - STATIC - produces output that is consistent as of a point in time
    - Locks held until list processing completes
  - STATIC QUIESCE - RECONs quiesced during list processing
    - No concurrent activity to the RECONs is allowed
      - Batch and utility jobs wait (even if not doing RECON activity)
    - Locking is not used in most cases
    - Consider using this option when static is required and many records will be listed (e.g. LIST.RECON)
- CHANGE.RECON LIST(STATIC|CONCURR)
  - Sets default for LIST commands

67

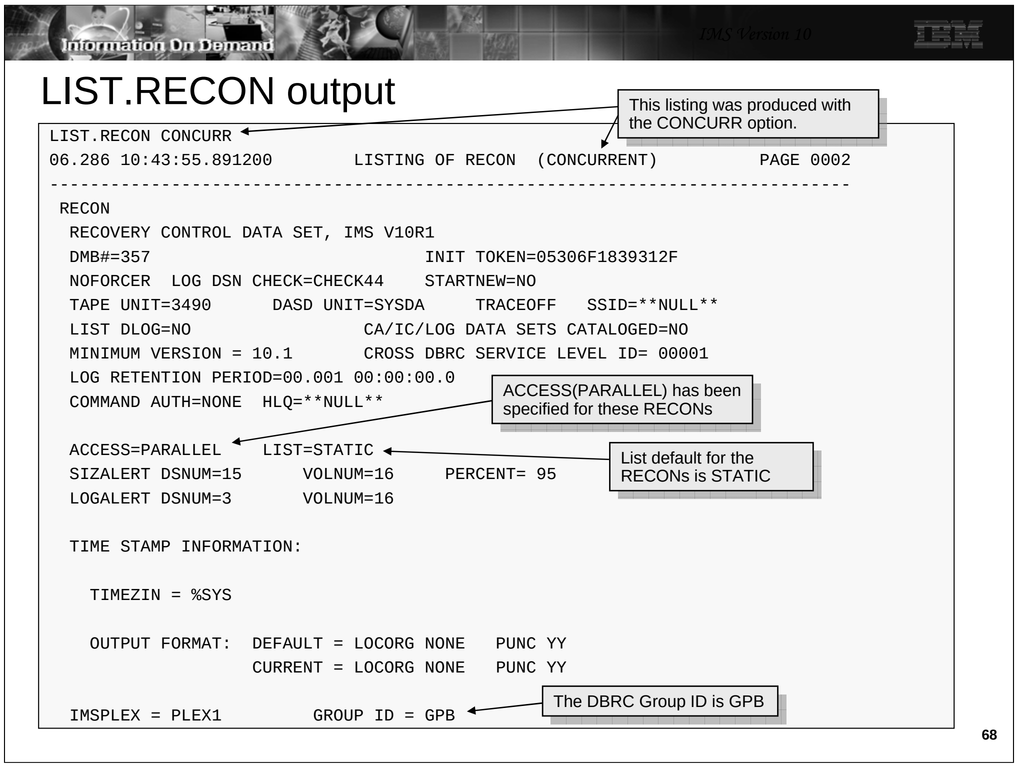
The LIST commands have new options with PRA. They are implemented with the CONCURR, STATIC, and QUIESCE keywords. These keywords are ignored when PRA is not being used.

If the CONCURR keyword is used, updates may occur to the RECONs while the list is being produced. This could make the output inconsistent since some of the records could be listed before a change and some after the change. For example, a database record might include an authorization to a subsystem but the subsystem record might not show the database as authorized to the subsystem. When CONCURR is used, a lock for a record is held only to read the record.

If the STATIC keyword is used, the listing is consistent. For example, if a database record listing includes an authorization to a subsystem, the subsystem record will show the database as authorized to the subsystem. When STATIC is used without the QUIESCE keyword, PRA locks each record when it reads it and holds the lock for the duration of the list processing.

The QUIESCE keyword may be used with STATIC. It quiesces all other activity to the RECONs during list processing. Other DBRC instances cannot access the RECONs during this time. Since all other activity is quiesced, locking is not needed to provide integrity or consistency unless there are retained locks. If there are retained locks from a failed DBRC for which recovery has not been done, a lock is held while DBRC is positioned on a record. QUIESCE reduces the overhead of the LIST command since it eliminates locking. You should consider using QUIESCE when a LIST command will read many RECON records. During the quiesce process batch jobs and utilities which use DBRC wait even if they are not attempting to access the RECONs. For example, batch jobs do no DL/I calls during the QUIESCE process.

The default of CONCURR or STATIC for the LIST command is set my the CHANGE.RECON LIST command. When RECONs are upgraded to V10, the default is set to STATIC. The QUIESCE keyword cannot be set by default.



LIST.RECON output

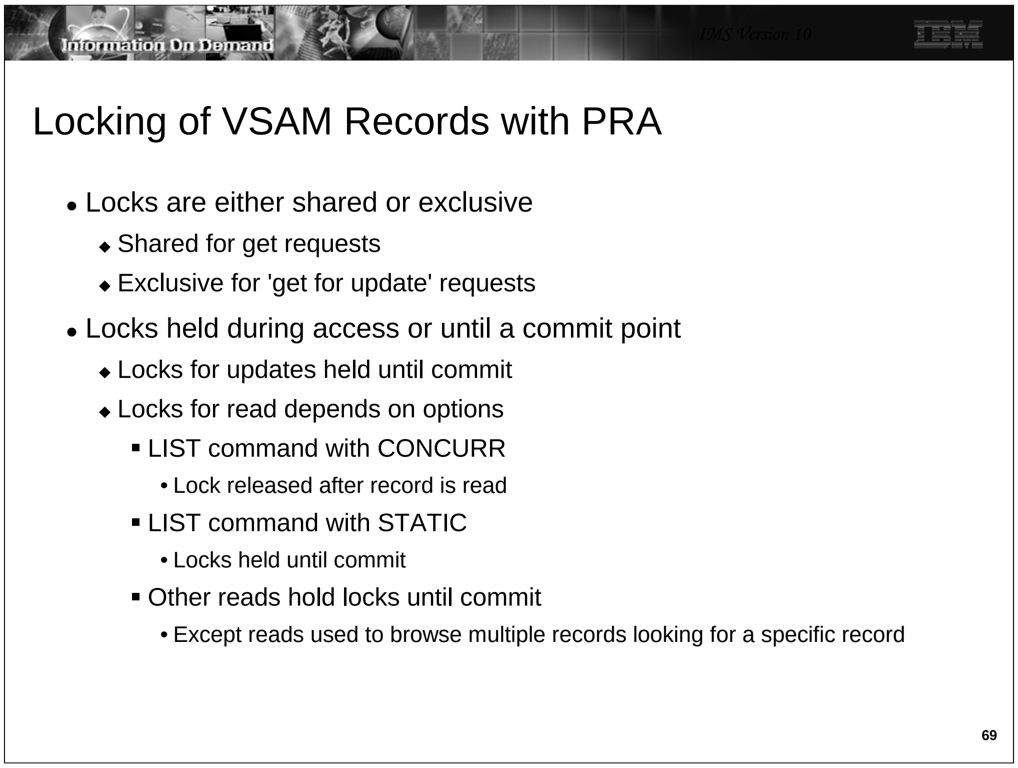
```
LIST.RECON CONCURR
06.286 10:43:55.891200        LISTING OF RECON  (CONCURRENT)          PAGE 0002
-------------------------------------------------------------------------------
 RECON
  RECOVERY CONTROL DATA SET, IMS V10R1
  DMB#=357                           INIT TOKEN=05306F1839312F
  NOFORCER  LOG DSN CHECK=CHECK44    STARTNEW=NO
  TAPE UNIT=3490      DASD UNIT=SYSDA     TRACEOFF   SSID=**NULL**
  LIST DLOG=NO                 CA/IC/LOG DATA SETS CATALOGED=NO
  MINIMUM VERSION = 10.1       CROSS DBRC SERVICE LEVEL ID= 00001
  LOG RETENTION PERIOD=00.001 00:00:00.0
  COMMAND AUTH=NONE  HLQ=**NULL**

  ACCESS=PARALLEL    LIST=STATIC
  SIZALERT DSNUM=15      VOLNUM=16     PERCENT= 95
  LOGALERT DSNUM=3       VOLNUM=16

  TIME STAMP INFORMATION:

    TIMEZIN = %SYS

    OUTPUT FORMAT:  DEFAULT = LOCORG NONE   PUNC YY
                    CURRENT = LOCORG NONE   PUNC YY

  IMSPLEX = PLEX1         GROUP ID = GPB
```

Information On Demand
IMS Version 10
IBM
This listing was produced with the CONCURR option.
ACCESS(PARALLEL) has been specified for these RECONs
List default for the RECONs is STATIC
The DBRC Group ID is GPB
68

This is an example of some of the output of a LIST.RECON command. The line which includes, "LISTING OF RECON (CONCURRENT)" indicates that this listing was produced with the CONCURR option. In this case, the LIST command included the CONCURR keyword.

Near the middle of the example there is a line of "ACCESS=PARALLEL   LIST=STATIC". This indicates that PRA is active for this RECON and that the default of LIST commands in STATIC.

At the bottom of the example is "GROUP ID = GPB". GPB is the DBRC group ID used for this set of RECONs.



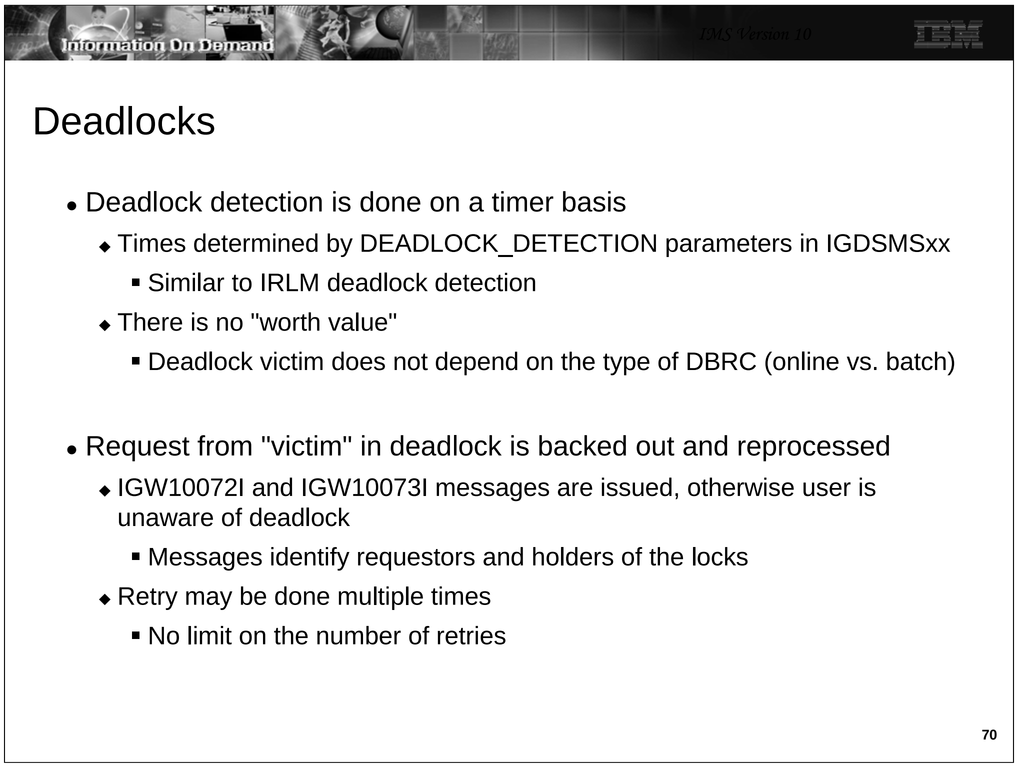
Locking of VSAM Records with PRA

- Locks are either shared or exclusive
  - Shared for get requests
  - Exclusive for 'get for update' requests
- Locks held during access or until a commit point
  - Locks for updates held until commit
  - Locks for read depends on options
    - LIST command with CONCURR
      - Lock released after record is read
    - LIST command with STATIC
      - Locks held until commit
    - Other reads hold locks until commit
      - Except reads used to browse multiple records looking for a specific record

PRA locks RECON records when they are read. The locks are either at a share level or an exclusive level. As the names suggest, multiple DBRC instances may hold the lock for a record at the same time if they are all at the share level. There can be only one holder of a lock for a record at the exclusive level.

VSAM get requests use share level locks. VSAM get for update requests use exclusive level locks.

Commit occurs when the DBRC request from IMS or the DBRC command has completed. Exclusive level locks are held until commit. Share level locks may be released before commit. Typically, share level locks are held until commit. When the CONCURR option is used for a LIST command, the locks are released as part of the read process. When the STATIC is used for a LIST command, the locks are held until commit. Sometimes DBRC browses RECON records of a type looking for a specific record. When this occurs, locks are obtained and released as each record is read. When the specific record is found, the lock is obtained again and held until commit.



Deadlocks

- Deadlock detection is done on a timer basis
  - Times determined by DEADLOCK_DETECTION parameters in IGDSMSxx
    - Similar to IRLM deadlock detection
  - There is no "worth value"
    - Deadlock victim does not depend on the type of DBRC (online vs. batch)
- Request from "victim" in deadlock is backed out and reprocessed
  - IGW10072I and IGW10073I messages are issued, otherwise user is unaware of deadlock
    - Messages identify requestors and holders of the locks
  - Retry may be done multiple times
    - No limit on the number of retries

70

The deadlock detection process of SMSVSAM is similar to that done by IRLM. There is a local cycle and a global cycle. During the local cycle deadlocks within a SMSVSAM are detected. During the global cycle deadlocks between units of recovery using different SMSVSAMs are detected. The local cycle uses a waiters list in the detection process. At each cycle a list is built of those units of recovery which are waiting. If the same wait exists in two successive cycles, the locks are examined to see if a deadlock exists. When a deadlock is found, a victim is chosen. Unlike the IRLM implementation, there is no "worth" value associated with units of recovery. A victim is chosen without regard to the type of DBRC instance in which it is executing.

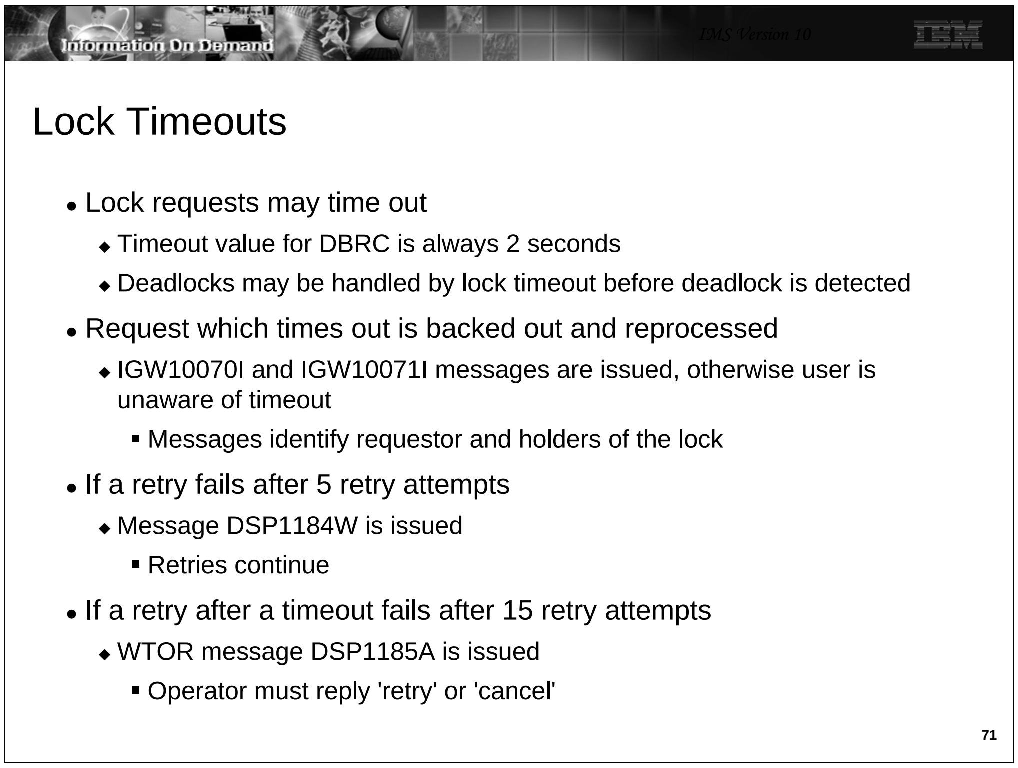
The UOR which is the victim in a deadlock is backed out and its request is reprocessed.

If a deadlock occurs an IGW10072I message is sent to the console. It is followed by IGW10073I messages. An IGW10073I message is issued once for each DBRC in the deadlock chain and includes the record that that DBRC holds and which record it is waiting for. The messages are:

IGW10072I *jobname stepname urid* VSAM RLS DETECTED A DEADLOCK. THERE ARE *nn* UNITS OF RECOVERY IN THE DEADLOCK CHAIN

IGW10073I {UNIT OF RECOVERY *urid* | SUBSYSTEM NAME *subsys* TRANSACTION ID *tranid*} RUNNING IN JOB *jobname* HOLDS {ADD TO END LOCK | EXCLUSIVE LOCK ON KEY | SHARED LOCK ON KEY} ON BASE CLUSTER NAME *cluster* [ALTERNATE INDEX NAME *altindex*] AND IS WAITING FOR {ADD TO END LOCK | EXCLUSIVE LOCK ON KEY | SHARED LOCK ON KEY} ON BASE CLUSTER NAME *cluster2* [ALTERNATE INDEX NAME *altindex2*] [HOLDING KEY VALUE = *key1* WAITING KEY VALUE = *key2*]

Deadlocks are retried indefinitely. This is no limit to the number of retries for them.



Lock Timeouts

- Lock requests may time out
  - Timeout value for DBRC is always 2 seconds
  - Deadlocks may be handled by lock timeout before deadlock is detected
- Request which times out is backed out and reprocessed
  - IGW10070I and IGW10071I messages are issued, otherwise user is unaware of timeout
    - Messages identify requestor and holders of the lock
- If a retry fails after 5 retry attempts
  - Message DSP1184W is issued
    - Retries continue
- If a retry after a timeout fails after 15 retry attempts
  - WTOR message DSP1185A is issued
    - Operator must reply 'retry' or 'cancel'

71

If a time out occurs an IGW10070I message is sent to the console. It is followed by an IGW10071I message for each holder of the lock. The messages are:

IGW10070I *jobname stepname urid* A REQUEST TIMED OUT WAITING FOR A LOCK. THERE ARE *nn* UNITS OF RECOVERY HOLDING THIS LOCK.

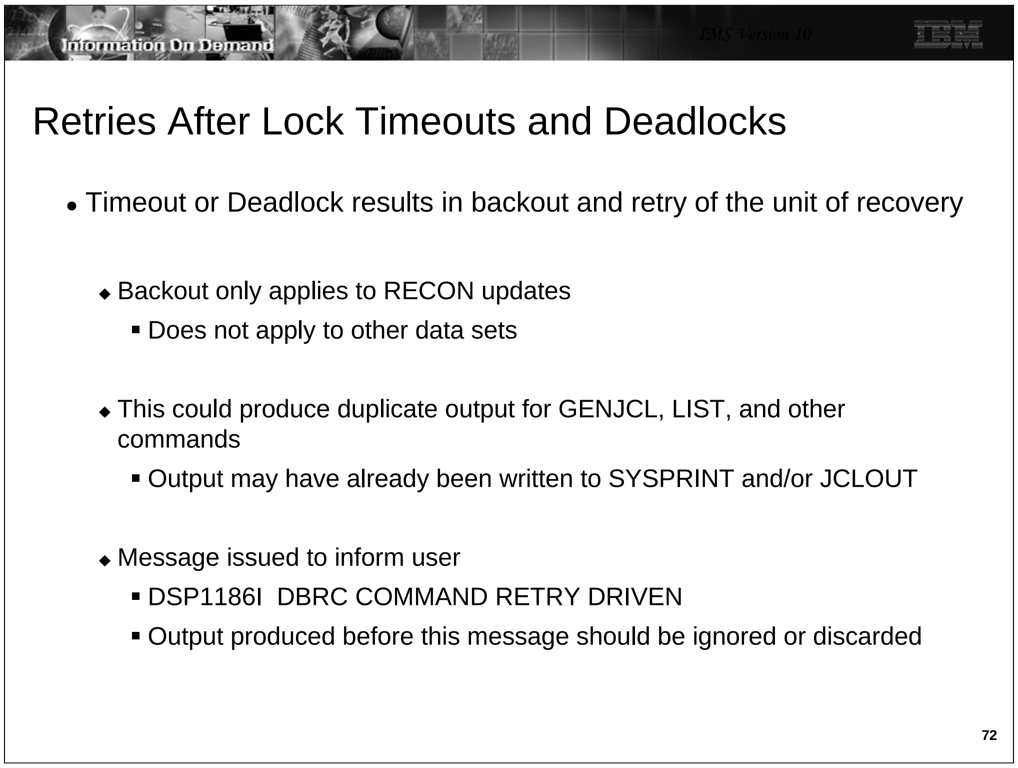
IGW10071I {UNIT OF RECOVERY *urid* | SUBSYSTEM NAME *subsys* TRANSACTION ID *tranid*} RUNNING IN JOB *jobname* HOLDS {ADD TO END LOCK | EXCLUSIVE LOCK ON KEY | SHARED LOCK ON KEY} IN BASE CLUSTER NAME *cluster* [PATH NAME *path*] CAUSING {TRUE | FALSE} CONTENTION. [KEY VALUE = *key*]

A random delay is done before a retry after a timeout. This is done to resolve situations where two requestors wait on each other multiple times

Message DSP1184W is sent to indicate that five retries have been attempted. Retries continue. No action is required by the operator. The text of the message is: `DSP1184W VSAM ACCESS ERROR ENCOUNTERED  5 TIMES RC=0008 RSN=0022`

If a timeout for the same lock request occurs 15 times for a unit of recovery, the DSP1185A message is sent. It is sent after the backout of the UOR. The message is a WTOR. The operator must reply to the message. The operator may be able to determine which DBRC instance is holding the lock. If so, the RETRY reply may be issued after the holder of the lock has completed its work. The text of the message is `DSP1185A VSAM ACCESS ERROR ENCOUNTERED 15 TIMES RC=0008 RSN=0022 - REPLY ’RETRY’ OR ’CANCEL’`

The RSN=0022 in the DSP1184W and DSP1185A messages indicate the reason for the error is a timeout for a lock request.


Information On Demand
IMS Version 10

Retries After Lock Timeouts and Deadlocks

- Timeout or Deadlock results in backout and retry of the unit of recovery
  - Backout only applies to RECON updates
    - Does not apply to other data sets
  - This could produce duplicate output for GENJCL, LIST, and other commands
    - Output may have already been written to SYSPRINT and/or JCLOUT
  - Message issued to inform user
    - DSP1186I DBRC COMMAND RETRY DRIVEN
    - Output produced before this message should be ignored or discarded

72

As explained before, a lock timeout or a deadlock results in the backout of the unit of recovery and the reprocessing of the DBRC request. The backout applies to the updates to the RECONs. It does not apply to any other output data sets, such as SYSPRINT or the JCLOUT data set used for GENJCL commands. Timeouts and deadlocks may produce duplicate outputs to these data sets. DBRC recognizes the possibility of these occurrences. It issues the DSP1186I message when a command is retried. This warns the user that there may be duplicate output from the command. Duplicate output tends to be a problem with GENJCL and LIST commands. It could occur for other commands, but the consequences are less serious. For example, the SYSPRINT output of a CHANGE.DB command could contain repeated lines.

The explanation of the DSP1186I message is:

> An error that could be retried was detected. DBRC is attempting to reprocess a command that might have produced external output. DBRC retries the processing, which encountered errors, that it considers capable of being retried (for example, deadlock or timeout). In this instance, data might have been written to a data set (for example, SYSPRINT) or the JCLOUT data sets. Since command processing is redriven, output produced prior to this message should be ignored. Output to a JCLOUT data set that is submitted directly to an internal reader may produce duplicate JCL or JCL that does not run.

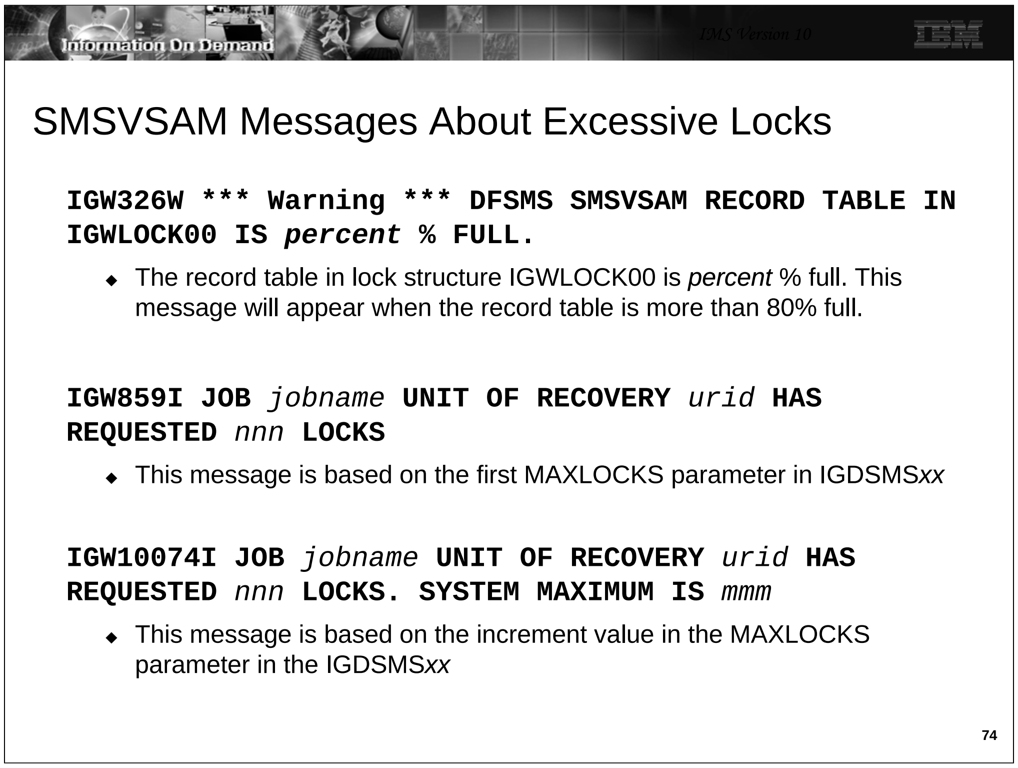
Monitoring of Locks

- The D SMS,CFLS command provides locking information

```
-D SMS,CFLS
IEE932I 236
IGW320I 14:52:52 Display SMS,CFLS
PRIMARY STRUCTURE:IGWLOCK00 VERSION:B8FC73F0F2A13441 SIZE:16384K
RECORD TABLE ENTRIES:48525 USED:4
System   Interval   LockRate   ContRate  FContRate   WaitQLen
14:52:52 Display SMS,CFLS
SC64     1 Minute        3.3     25.969      0.000       0.33
SC64      1 Hour        84.0      4.797      0.416       0.10
SC64      8 Hour        17.8      1.004      0.092       0.00
SC64       1 Day         7.7      0.354      0.044       0.00
  (03)   1 Minute        2.2     17.714      0.000       0.19
  (03)    1 Hour        56.0      3.220      0.282       0.05
  (03)    8 Hour        11.4      0.671      0.059       0.00
  (03)     1 Day         4.5      0.236      0.024       0.00
***************** LEGEND ******************
LockRate = number of lock requests per second
CONTRATE = % of lock requests globally managed
FCONTRATE = % of lock requests falsely globally managed
WaitQLen = Average number of requests waiting for locks
```

The D SMS,CFLS command displays locking information from SMSVSAM. In this example the lock structure size is 16384K or 16M. This implies that the lock table size is 8M which is half of 16M. There are 48252 record table (or record list) entries. Only 4 of these record table entries are being used at the time of the display.

The following lines show the locking rate, the contention rate, the false contention rate, and the average number of requests waiting for locks during the last minute, the last hour, the last 8 hours, and the last day. The data returned by the command is for the system where the command is issued and an average of all of the systems. The response shows the lock and contention rates for system SC64. The lines beginning with "(03)" are averages for all of the systems. The "(03)" indicates that there are three systems.



SMSVSAM Messages About Excessive Locks

IGW326W * Warning *** DFSMS SMSVSAM RECORD TABLE IN IGWLOCK00 IS *percent* % FULL.**

- The record table in lock structure IGWLOCK00 is *percent* % full. This message will appear when the record table is more than 80% full.

IGW859I JOB *jobname* **UNIT OF RECOVERY** *urid* **HAS REQUESTED** *nnn* **LOCKS**

- This message is based on the first MAXLOCKS parameter in IGDSMS*xx*

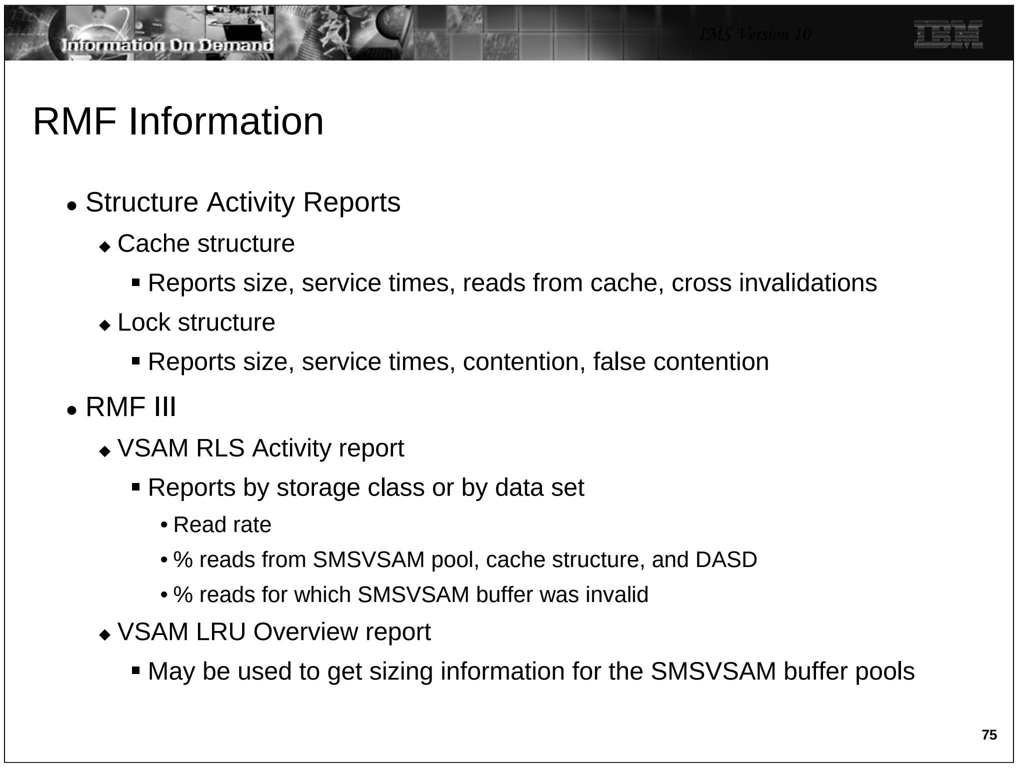
IGW10074I JOB *jobname* **UNIT OF RECOVERY** *urid* **HAS REQUESTED** *nnn* **LOCKS. SYSTEM MAXIMUM IS** *mmm*

- This message is based on the increment value in the MAXLOCKS parameter in the IGDSMS*xx*

The IGW326W message is issued when the record table (record list) in the lock structure is 80% full. If the record table becomes full, new locks for updates cannot be granted.

The IGW859I message is issued when a unit of recovery first exceeds the first MAXLOCKS subparameter in the IGDSMSxx member.

The IGW10074I message is issued after the IGW859I message when the unit of recovery requests more locks. Its issuance is controlled by the second MAXLOCKS subparameter. This subparameter specifies an incremental number of locks. The message is issued each time the incremental number of additional locks is requested.



RMF Information

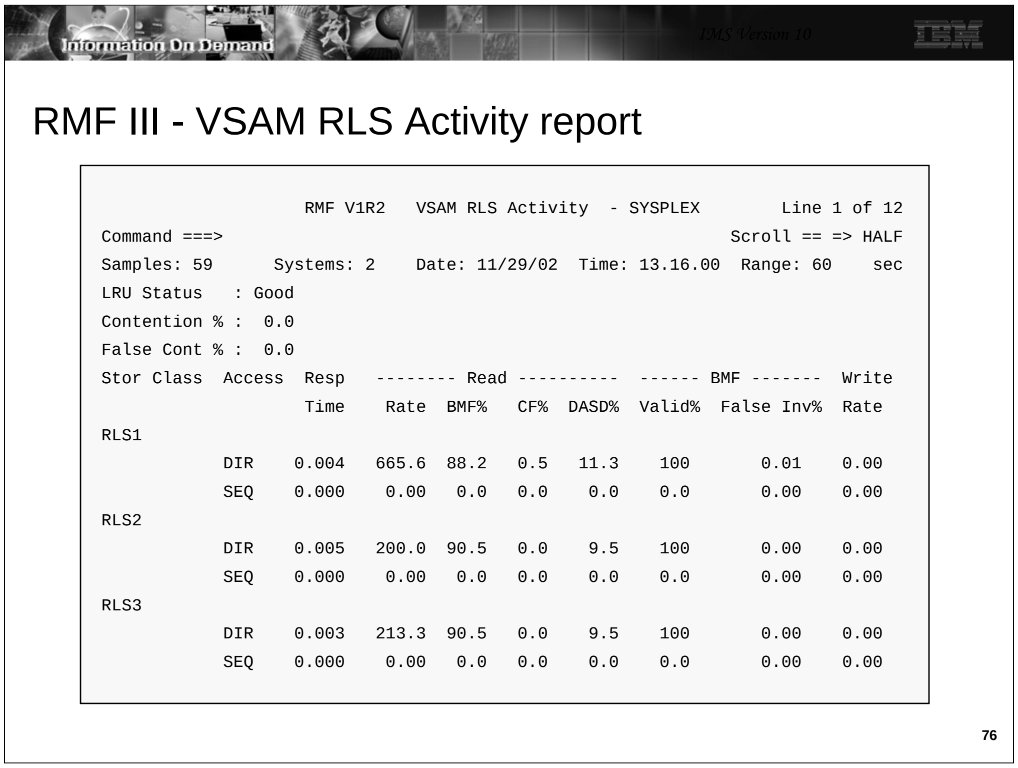
- Structure Activity Reports
  - Cache structure
    - Reports size, service times, reads from cache, cross invalidations
  - Lock structure
    - Reports size, service times, contention, false contention
- RMF III
  - VSAM RLS Activity report
    - Reports by storage class or by data set
      - Read rate
      - % reads from SMSVSAM pool, cache structure, and DASD
      - % reads for which SMSVSAM buffer was invalid
  - VSAM LRU Overview report
    - May be used to get sizing information for the SMSVSAM buffer pools

75

Lock and cache structure usage information is also available from the RMF Structure Activity Report. IMS data sharing users are familiar with these reports for the IRLM lock structure and OSAM and VSAM cache structures. The same type of report is available for the SMSVSAM structures.

RMF III has a VSAM RLS Activity report. An example of this report is shown on the next page.

The RMF III VSAM LRU Overview report provides information about the size of the SMSVSAM buffer pool. The size of this pool is dynamically adjusted by SMSVSAM. The RLS_MAX_POOL_SIZE parameter sets the limit for the pool size, although it might be temporarily exceeded. During each LRU cycle, SMSVSAM determines whether the system is over the goal. If it is, adjustments are made. The report shows the percentage of time when the buffer pool exceeded the desired limit. It also reports on percentages of requests which were satisfied from the buffer pool, the cache structures, and DASD.

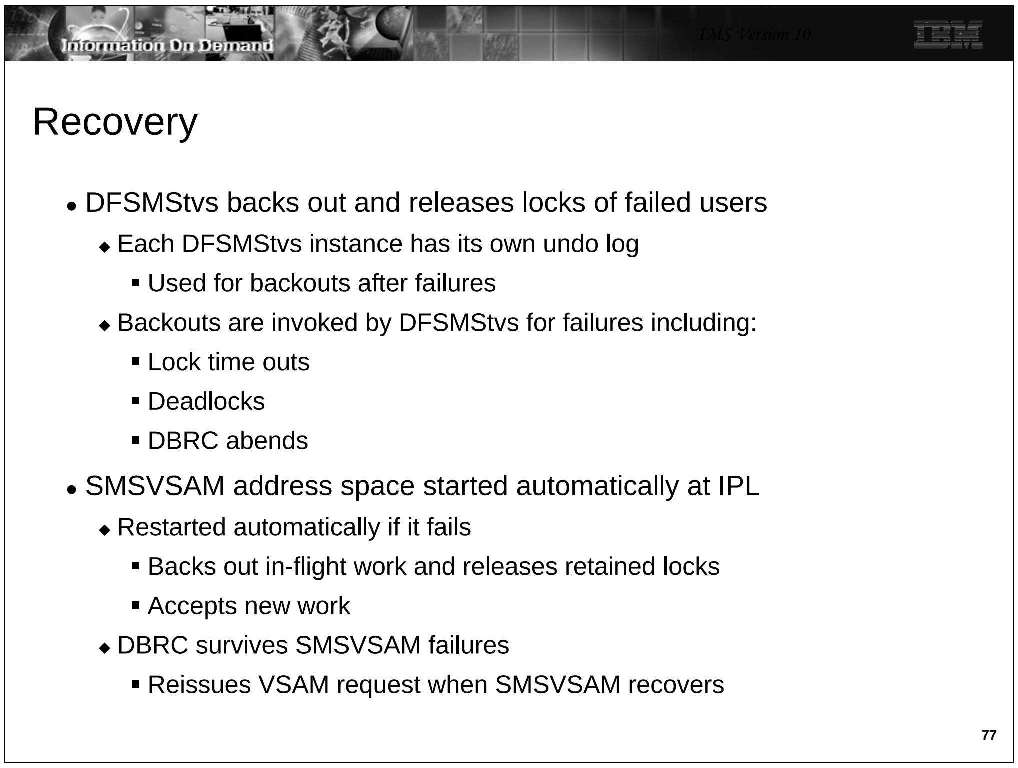


RMF III - VSAM RLS Activity report

```
                    RMF V1R2   VSAM RLS Activity  - SYSPLEX        Line 1 of 12
Command ===>                                                   Scroll == => HALF
Samples: 59      Systems: 2    Date: 11/29/02  Time: 13.16.00  Range: 60    sec
LRU Status   : Good
Contention % :  0.0
False Cont % :  0.0
Stor Class  Access  Resp   -------- Read ----------  ------ BMF -------  Write
                    Time    Rate  BMF%   CF%  DASD%  Valid%  False Inv%  Rate
RLS1
            DIR    0.004   665.6  88.2   0.5   11.3    100       0.01    0.00
            SEQ    0.000    0.00   0.0   0.0    0.0    0.0       0.00    0.00
RLS2
            DIR    0.005   200.0  90.5   0.0    9.5    100       0.00    0.00
            SEQ    0.000    0.00   0.0   0.0    0.0    0.0       0.00    0.00
RLS3
            DIR    0.003   213.3  90.5   0.0    9.5    100       0.00    0.00
            SEQ    0.000    0.00   0.0   0.0    0.0    0.0       0.00    0.00
```

76

This is an example of the RMF III VSAM RLS Activity report. It reports on three storage classes used by RLS. These storage classes are RLS1, RLS2, and RLS3. For each class it reports direct and sequential access activities. Under the "Read" columns it reports the percentage of requests that are satisfied from the SMSVSAM buffers (BMF%), from the coupling facility structures (CF%), and from DASD (DASD%). Under "BMF" is shows the percentage of reads which found the CI in the SMSVSAM buffer pool but the buffer had been invalidated by an update to the CI in another system.



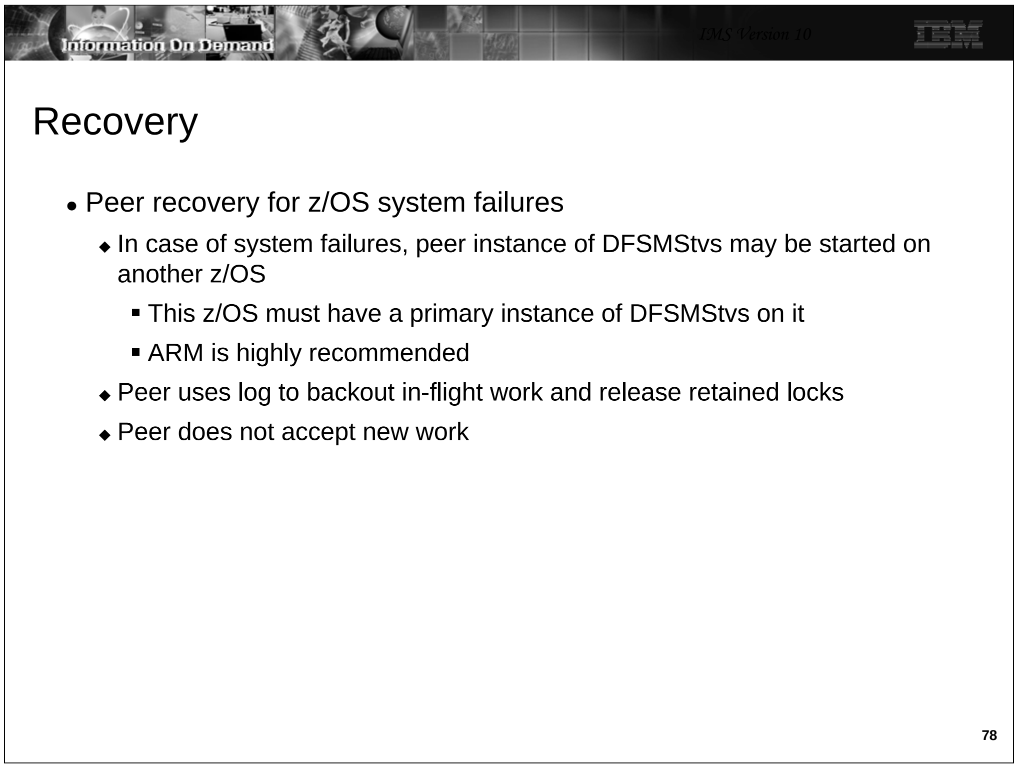
Recovery

- DFSMStvs backs out and releases locks of failed users
  - Each DFSMStvs instance has its own undo log
    - Used for backouts after failures
  - Backouts are invoked by DFSMStvs for failures including:
    - Lock time outs
    - Deadlocks
    - DBRC abends
- SMSVSAM address space started automatically at IPL
  - Restarted automatically if it fails
    - Backs out in-flight work and releases retained locks
    - Accepts new work
  - DBRC survives SMSVSAM failures
    - Reissues VSAM request when SMSVSAM recovers

77

Each DFSMStvs instance has its own undo log and associated shunt log. The logs are shared by all users of the DFSMStvs instance, but they are not shared by DFSMStvs instances on different z/OS systems. The undo and shunt logs are used for backouts when there is a failure or deadlock by a user of DFSMStvs services, such as a DBRC instance.

When you IPL a z/OS system the SMSVSAM address space is automatically started if it has been defined for the system. If the address space fails, it is automatically restarted by z/OS. When it is restarted, it backs out any work that was in-flight at the time of the failure. It also releases the locks held for the work. After this recovery, it accepts new work.



Recovery

- Peer recovery for z/OS system failures
  - In case of system failures, peer instance of DFSMStvs may be started on another z/OS
    - This z/OS must have a primary instance of DFSMStvs on it
    - ARM is highly recommended
  - Peer uses log to backout in-flight work and release retained locks
  - Peer does not accept new work

78

Peer recovery is the process of completing the work that was left in an incomplete state due to the failure of an instance of DFSMStvs. Peer recovery is done by another instance of DFSMStvs. Peer recovery occurs only in cases of system failure, not merely when DFSMStvs fails. When a z/OS system fails, a peer recovery instance of DFSMStvs is used to back out in-flight work and release locks. The z/OS system where peer recovery is run must have a primary instance of DFSMStvs running on it.

Peer recovery is not automatically invoked unless Automatic Restart Management (ARM) is used to invoke it. Without ARM, an operator command is required to start peer recovery. ARM information is available in the *z/OS DFSMStvs Planning and Operating Guide* and in *z/OS MVS Setting Up a Sysplex*.

Peer recovery backs out the in-flight work of the failed system and releases the locks for this work. The peer does not accept new work.

The peer recovery instance of DFSMStvs does the following:

• It registers with VSAM RLS and RRS as the failed instance of DFSMStvs and indicates to RRS that it is beginning restart processing as the failed instance.

• It reads the failed instance's undo log to retrieve information about in-progress units of recovery.

• It invokes RRS to retrieve information about unit of recovery status.

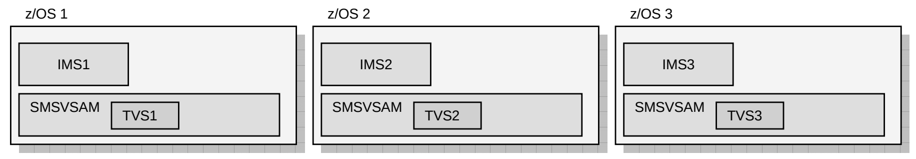
• It processes the log data and the information returned by RRS to determine whether to commit or back out units of recovery. It backs out in-flight work and releases locks for this work.

• When all outstanding units of recovery have been processed, it unregisters with RRS and VSAM RLS.

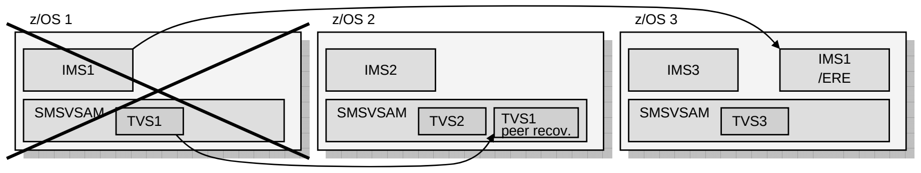
The peer recovery instance of DFSMStvs registers as the failed instance in order to gain access to the failed instance's resources. This registration persists until peer recovery is complete. As a result, should the failed instance restart, it will be unable to initialize because its attempt to register will fail. It may be necessary to restart DFSMStvs manually once peer recovery is complete.

Peer Recovery Example

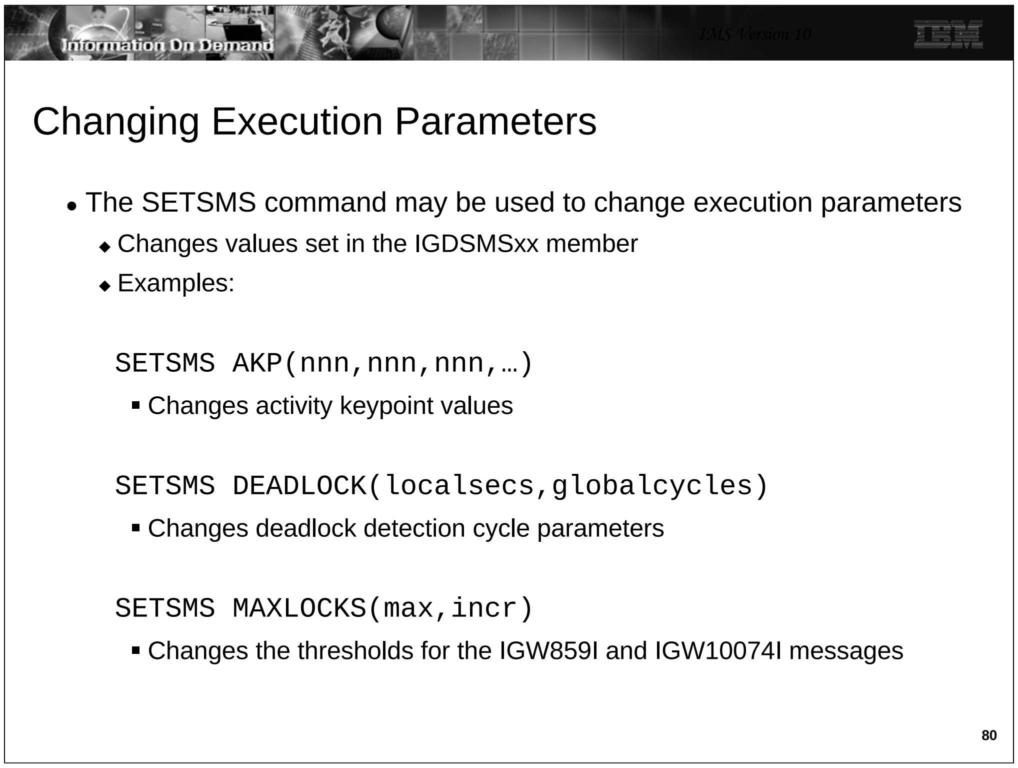
- Before failure:



- After failure:
  - TVS1 peer recovery may be done on z/OS2 or z/OS3
    - It does not require that IMS1 be restarted
  - IMS1 may be restarted on a different z/OS



This example illustrates that peer recovery may be done on any z/OS system which has a SMSVSAM address space with a current primary instance of DFSMStvs. Peer recovery does not require that the IMSs using it be restarted. Of course, you would probably want to restart any failed IMS online system. The emergency restart of IMS1 could be done on any z/OS which has a DFSMStvs instance. It does not have to be done on the same z/OS system where peer recovery is being done for the failed instance of DFSMStvs.



Changing Execution Parameters

- The SETSMS command may be used to change execution parameters
  - Changes values set in the IGDSMSxx member
  - Examples:

```
SETSMS AKP(nnn,nnn,nnn,…)
```

- Changes activity keypoint values

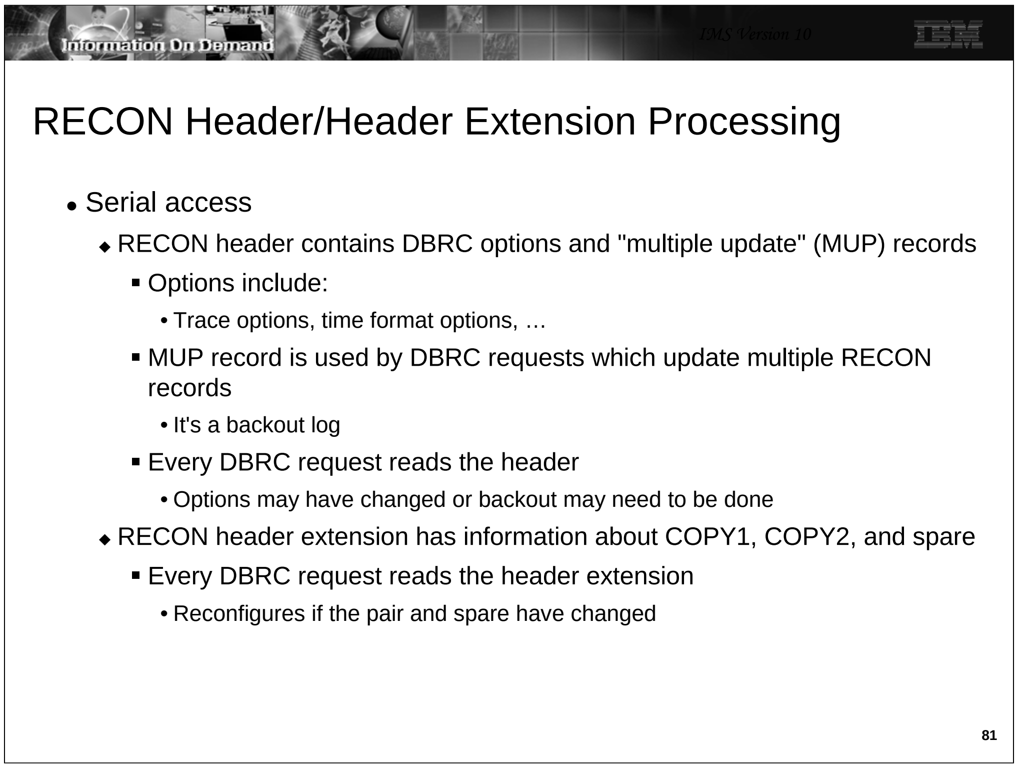
```
SETSMS DEADLOCK(localsecs,globalcycles)
```

- Changes deadlock detection cycle parameters

```
SETSMS MAXLOCKS(max,incr)
```

- Changes the thresholds for the IGW859I and IGW10074I messages

The SETSMS command may be used to change parameters that were specified in the IGDSMSxx member. Examples of the kinds of parameters that may be changed are shown here.



RECON Header/Header Extension Processing

- Serial access
  - RECON header contains DBRC options and "multiple update" (MUP) records
    - Options include:
      - Trace options, time format options, …
    - MUP record is used by DBRC requests which update multiple RECON records
      - It's a backout log
    - Every DBRC request reads the header
      - Options may have changed or backout may need to be done
  - RECON header extension has information about COPY1, COPY2, and spare
    - Every DBRC request reads the header extension
      - Reconfigures if the pair and spare have changed

81

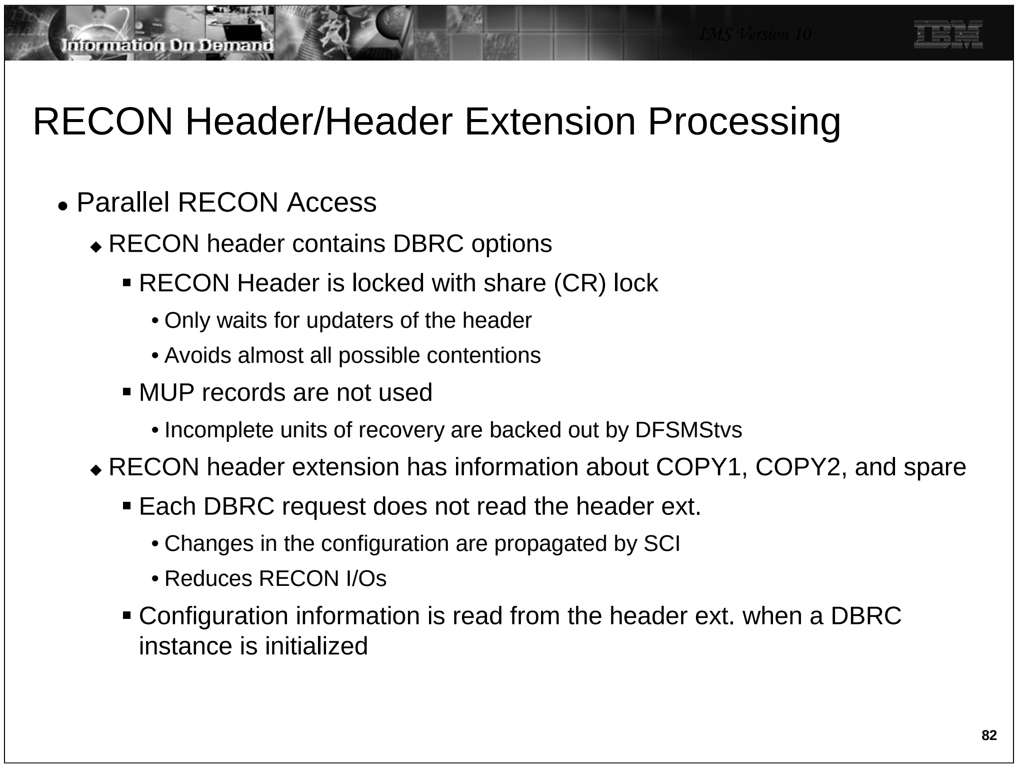
This explains how the RECON header and header extension records are used when Parallel RECON Access is NOT used.

With serial access, the RECON header and header extension are read by each DBRC request.

The header is read for two reasons. It is read to determine the current RECON options. These include things such as trace options and time format options. Second, the header is read to determine is there is a multiple update (MUP) record. A MUP record is written before DBRC makes updates to multiple RECON records as part of one DBRC request. This is used to ensure that all of the updates are made, or if they are not made, that they may be backed out. MUP records contain information sufficient for backing out updates to the RECONs. After the MUP record is written, DBRC makes the updates to the indicated RECON records. When these are completed, the MUP record is deleted. Each DBRC request first checks for a MUP record. If one exists, it indicates that another DBRC instance has failed in the middle of making multiple updates. The DBRC instance which reads the MUP record, backs out the partial updates and deletes the MUP record. This mechanism provides integrity with serial access to the RECONs.

The header extension is read to determine the current RECON configuration. This is the current COPY1, COPY2 and spare data sets. They could have been changed by another DBRC instance since this DBRC last accessed the RECONs.

This processing differs with Parallel RECON Access.



RECON Header/Header Extension Processing

- Parallel RECON Access
  - RECON header contains DBRC options
    - RECON Header is locked with share (CR) lock
      - Only waits for updaters of the header
      - Avoids almost all possible contentions
    - MUP records are not used
      - Incomplete units of recovery are backed out by DFSMStvs
  - RECON header extension has information about COPY1, COPY2, and spare
    - Each DBRC request does not read the header ext.
      - Changes in the configuration are propagated by SCI
      - Reduces RECON I/Os
    - Configuration information is read from the header ext. when a DBRC instance is initialized

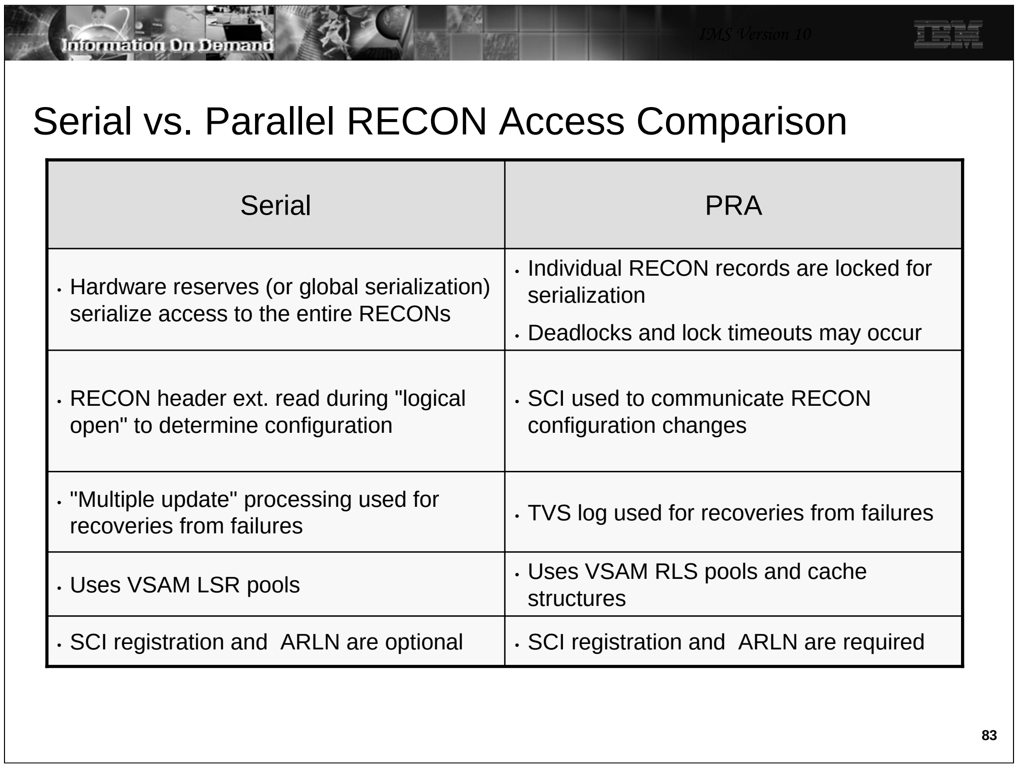
The header record is read by DBRC "logical open" but is locked with a shared lock except when updates are done. It is read to check on the current RECON options. Updates are rare. They are typically done by a DBRC CHANGE.RECON command.

Parallel RECON Access eliminates the need to read the RECON header extension for every DBRC request.

The RECON header extension still contains the current RECON configuration information, but DBRC does not need to read the header for every DBRC request. If a DBRC instance reconfigures the RECONs, it uses SCI to communicate the new configuration to the other instances. The configuration information is obtained from the header extension when a new DBRC instance is initialized. The header extension does not have to be read for every subsequent request of the DBRC instance.

MUP records are not used with Parallel RECON Access. Backouts of incomplete updates are done by DFSMStvs using its undo log stream. This does not eliminate the need to read the header for each DBRC request. The continues to be read to find any changes to RECON options. (e.g. if a CHANGE.RECON TRACEON has been issued).

These changes in processing of the header extension record with Parallel RECON Access eliminate a reason for possible contention on the header extension record. It also reduces the number of I/Os to the RECONs required for most DBRC requests.



Serial vs. Parallel RECON Access Comparison

Serial	PRA
• Hardware reserves (or global serialization) serialize access to the entire RECONs	• Individual RECON records are locked for serialization • Deadlocks and lock timeouts may occur
• RECON header ext. read during "logical open" to determine configuration	• SCI used to communicate RECON configuration changes
• "Multiple update" processing used for recoveries from failures	• TVS log used for recoveries from failures
• Uses VSAM LSR pools	• Uses VSAM RLS pools and cache structures
• SCI registration and ARLN are optional	• SCI registration and ARLN are required

83

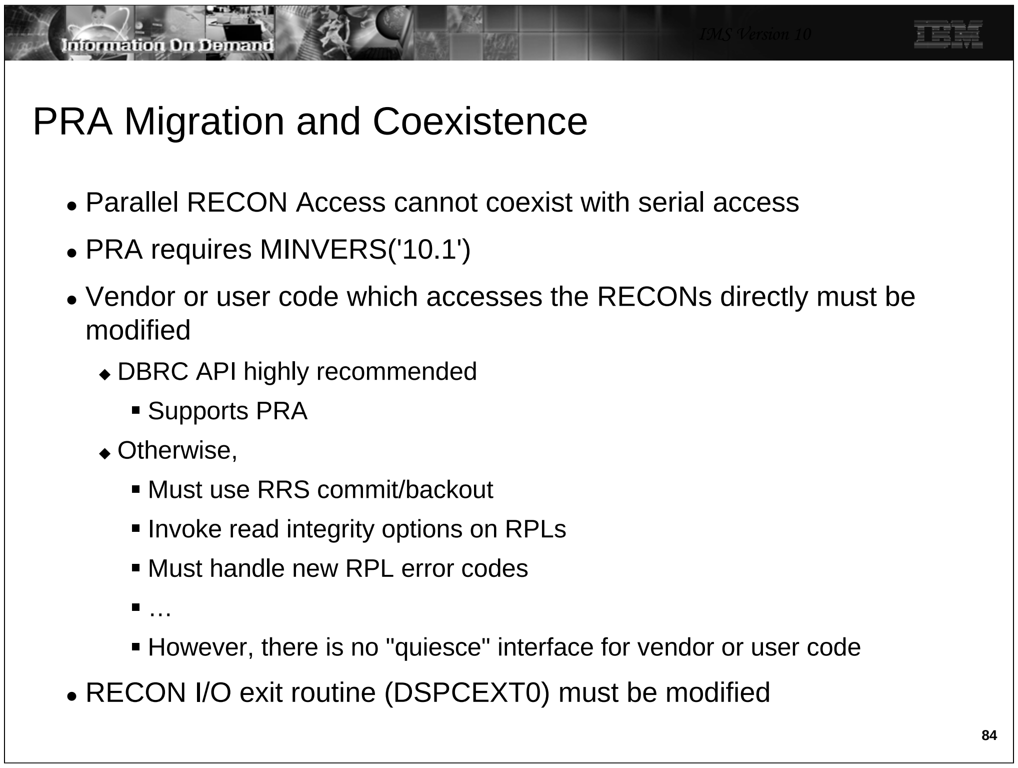
Serial processing uses hardware reserves or global data set serialization to serialize access to the RECONs. Only one DBRC instance may be accessing the RECONs at any time. PRA allows multiple DBRC instances to access the RECONs at the same time. It locks individual RECON records to provide integrity. Since it uses locking, there is the possibility of deadlocks or timeouts for lock requests.

Serial processing maintains the status of the RECONs in the header/header extension records. Every access to the RECONs first has to check the header/header extension for the current status. PRA uses SCI communication to notify other DBRC instances of any change in the RECON configuration. PRA does not access the header/header extension with every access to the RECONs.

Serial processing has special handling for requests that require updates to multiple RECON records. This is needed to maintain integrity for multiple updates. It writes a MUP record to the header/header extension before doing its intended multiple updates and then deletes the MUP information after the intended multiple updates are done. PRA does not have this processing. It uses TVS to maintain integrity for multiple updates. If one or more RECON records are updated, but the other records in the request cannot be updated, the TVS log is used to back out the updates that were made.

Serial processing uses VSAM LSR (local shared resources) pools. Each DBRC instance has its own pools. These pools are invalidated at the beginning of each request to DBRC. This is necessary because another DBRC instance may have modified the CIs that are in the pool. PRA does not use LSR pools. It uses the SMSVSAM RLS pool in its LPAR. Parallel Sysplex buffer invalidations are used to invalidate only the buffers that have been updated by DBRC instances running in other LPARs. Cache structures are used in the invalidation process. These cache structures also maintain copies of RECON CIs so that some reads to the RECON data sets may be eliminated.

SCI registration and Automatic RECON Loss Notification are optional with serial access. They are required with PRA.



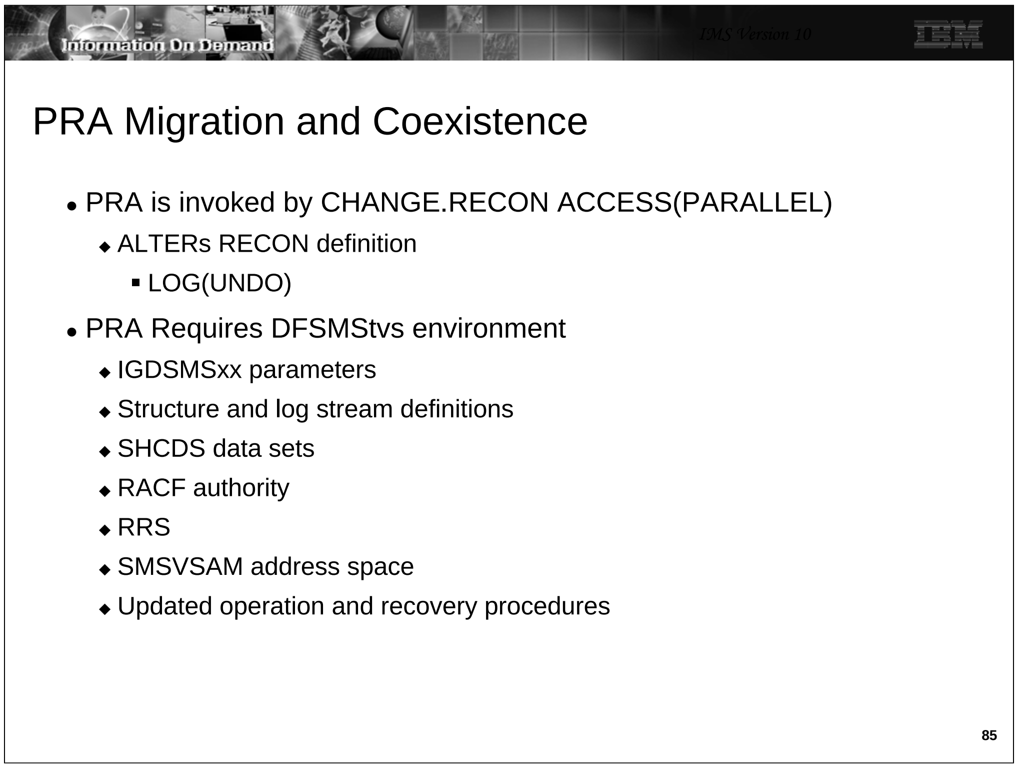
PRA Migration and Coexistence

- Parallel RECON Access cannot coexist with serial access
- PRA requires MINVERS('10.1')
- Vendor or user code which accesses the RECONs directly must be modified
  - DBRC API highly recommended
    - Supports PRA
  - Otherwise,
    - Must use RRS commit/backout
    - Invoke read integrity options on RPLs
    - Must handle new RPL error codes
    - …
    - However, there is no "quiesce" interface for vendor or user code
- RECON I/O exit routine (DSPCEXT0) must be modified

84

This is a summary of the migration and coexistence considerations. Most of these have been covered previously.

Products and application programs which access the RECONs natively must be adapted for DFSMStvs. The recommended way of handling this is the use of the DBRC API. The DBRC API supports both Parallel RECON Access and serial access. If the DBRC API is not used, the programs must support the DFSMStvs interfaces. These include the use of RRS for commit and backout, opening the RECONs for DFSMStvs access, and the use of RPLs which invoke DFSMStvs access. Even if they successfully implement these interfaces, full support cannot be done. For example, there is no interface to quiesce user code for the LIST command with the STATIC QUIESCE option.

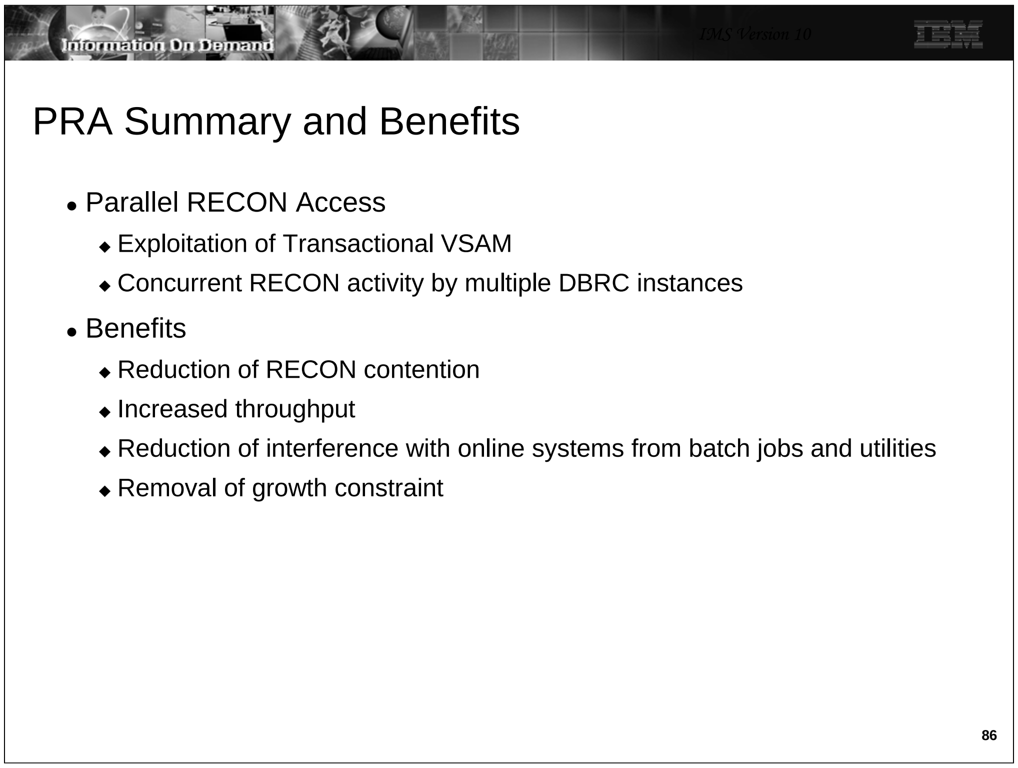


PRA Migration and Coexistence

- PRA is invoked by CHANGE.RECON ACCESS(PARALLEL)
  - ALTERs RECON definition
    - LOG(UNDO)
- PRA Requires DFSMStvs environment
  - IGDSMSxx parameters
  - Structure and log stream definitions
  - SHCDS data sets
  - RACF authority
  - RRS
  - SMSVSAM address space
  - Updated operation and recovery procedures

85

Parallel RECON Access is invoked with the CHANGE.RECON ACCESS(PARALLEL) command. The command internally issues an ALTER for the RECON data sets changing the LOG parameter to LOG(UNDO). This requires that the DFSMStvs environment be present. Of course, operations procedures and recovery procedures should be updated for DFSMStvs.

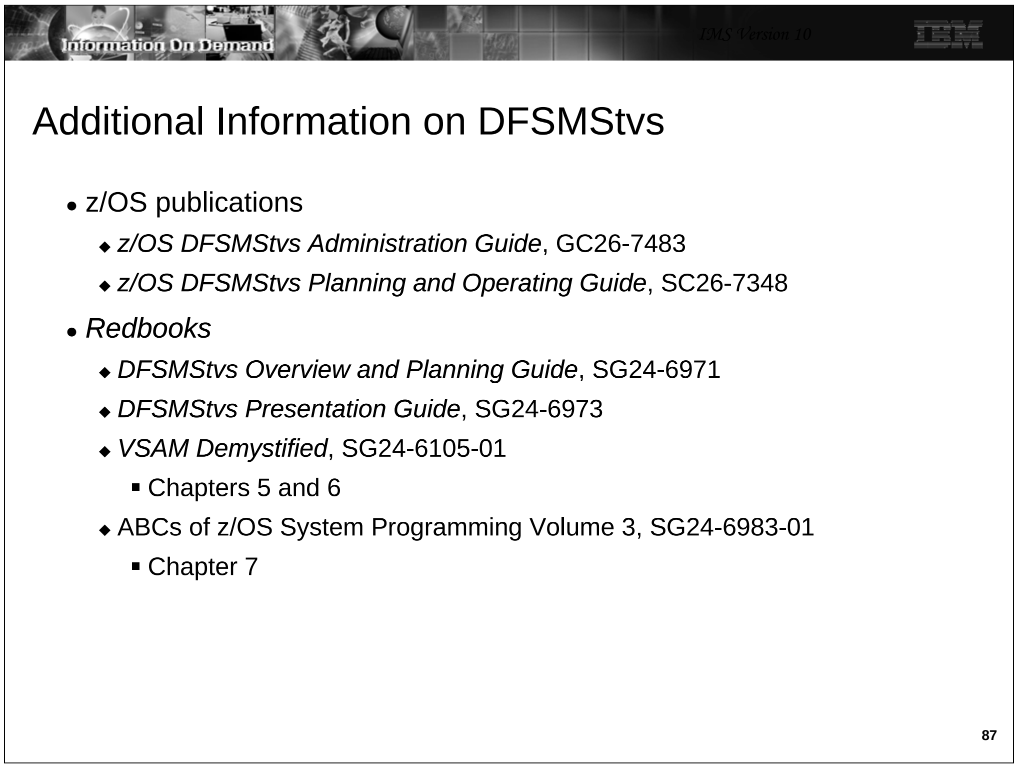

Information On Demand
IMS Version 10

PRA Summary and Benefits

- Parallel RECON Access
  - Exploitation of Transactional VSAM
  - Concurrent RECON activity by multiple DBRC instances
- Benefits
  - Reduction of RECON contention
  - Increased throughput
  - Reduction of interference with online systems from batch jobs and utilities
  - Removal of growth constraint

86

PRA exploits Transactional VSAM for the RECON data sets. This allows multiple DBRC instances, which are online systems, IMS batch jobs, and IMS utilities, to access the RECON data sets concurrently. Volume or global data set serialization is not required. This reduces RECON contention. The reduced contention provides for greater growth potential and increased throughput possibilities. The primary recipient of the benefits for most installations will be online systems since they will suffer less interference from batch jobs and utilities which are accessing the RECONs.

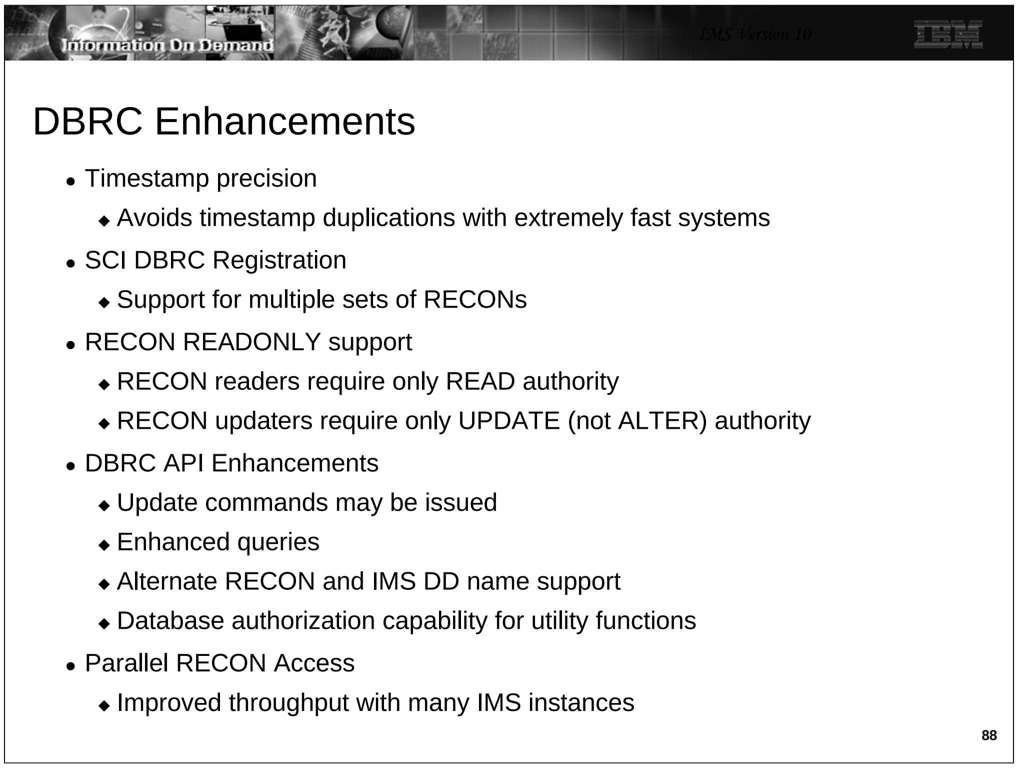


Additional Information on DFSMStvs

- z/OS publications
  - *z/OS DFSMStvs Administration Guide*, GC26-7483
  - *z/OS DFSMStvs Planning and Operating Guide*, SC26-7348
- *Redbooks*
  - *DFSMStvs Overview and Planning Guide*, SG24-6971
  - *DFSMStvs Presentation Guide*, SG24-6973
  - *VSAM Demystified*, SG24-6105-01
    - Chapters 5 and 6
  - ABCs of z/OS System Programming Volume 3, SG24-6983-01
    - Chapter 7

These books provide information on DFSMStvs. Much of the information assumes that DFSMStvs is being used to support concurrent access to VSAM files from CICS and batch programs. When reading these publications keep in mind that DBRC use of DFSMStvs has the following characteristics:

- Forward recovery logs are not used for the RECONs.
- Each DBRC request is a "transaction". Requests include:
  - DBRC commands or subsets of them. Some commands are processed as multiple requests.
  - Subsystem signons and signoffs
  - Database authorizations or unauthorizations
  - Recordings of log
  - Recordings of image copies
  - Recordings of reorganizations
  - Recordings of database recoveries
  - Creation of allocation records (first update to a data set by a subsystem)
  - and many others

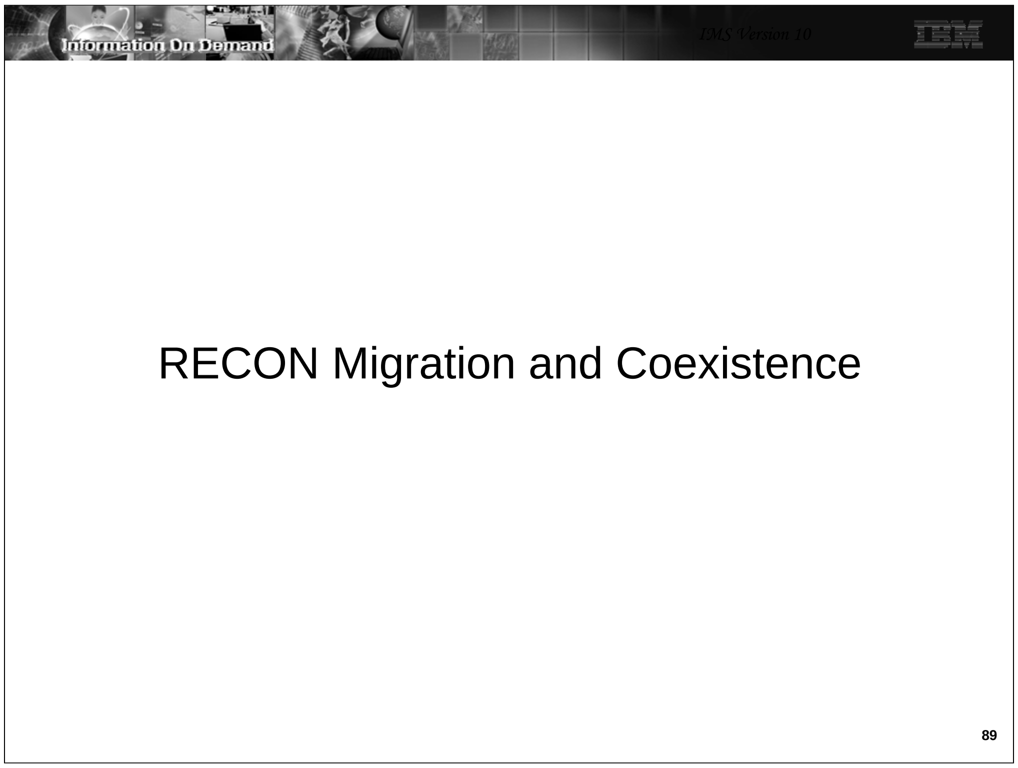

Information On Demand
IMS Version 10

DBRC Enhancements

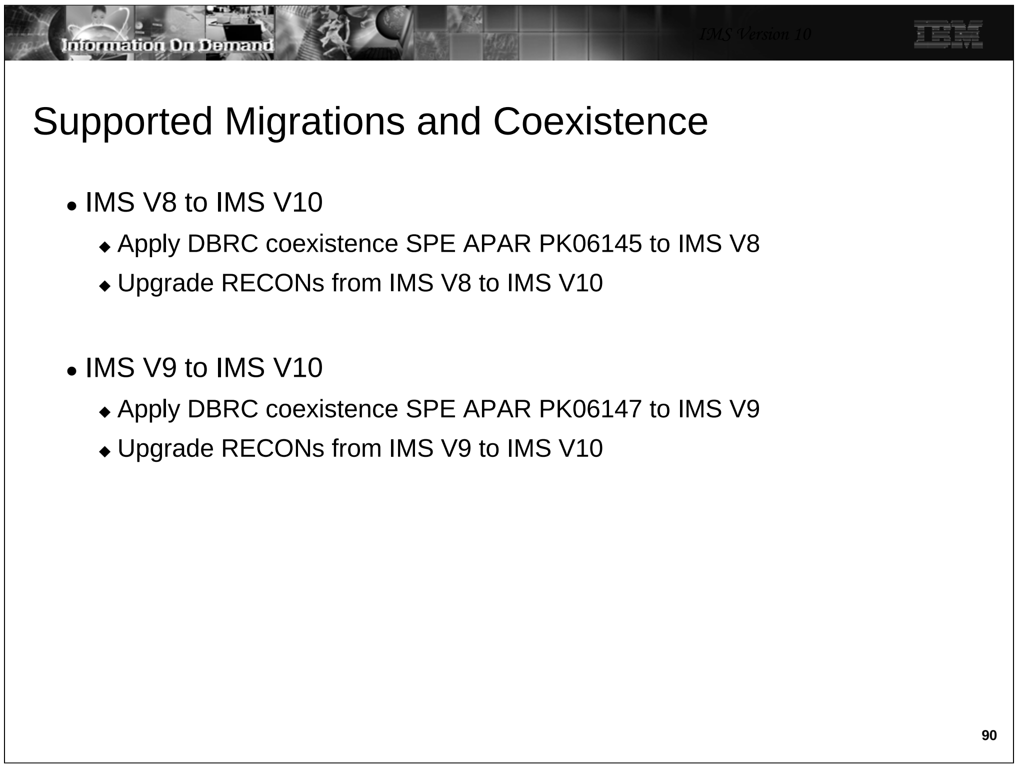
- Timestamp precision
  - Avoids timestamp duplications with extremely fast systems
- SCI DBRC Registration
  - Support for multiple sets of RECONs
- RECON READONLY support
  - RECON readers require only READ authority
  - RECON updaters require only UPDATE (not ALTER) authority
- DBRC API Enhancements
  - Update commands may be issued
  - Enhanced queries
  - Alternate RECON and IMS DD name support
  - Database authorization capability for utility functions
- Parallel RECON Access
  - Improved throughput with many IMS instances

88

This summarizes the DBRC enhancements in IMS V10.



RECON Migration and Coexistence



Supported Migrations and Coexistence

- IMS V8 to IMS V10
  - Apply DBRC coexistence SPE APAR PK06145 to IMS V8
  - Upgrade RECONs from IMS V8 to IMS V10
- IMS V9 to IMS V10
  - Apply DBRC coexistence SPE APAR PK06147 to IMS V9
  - Upgrade RECONs from IMS V9 to IMS V10

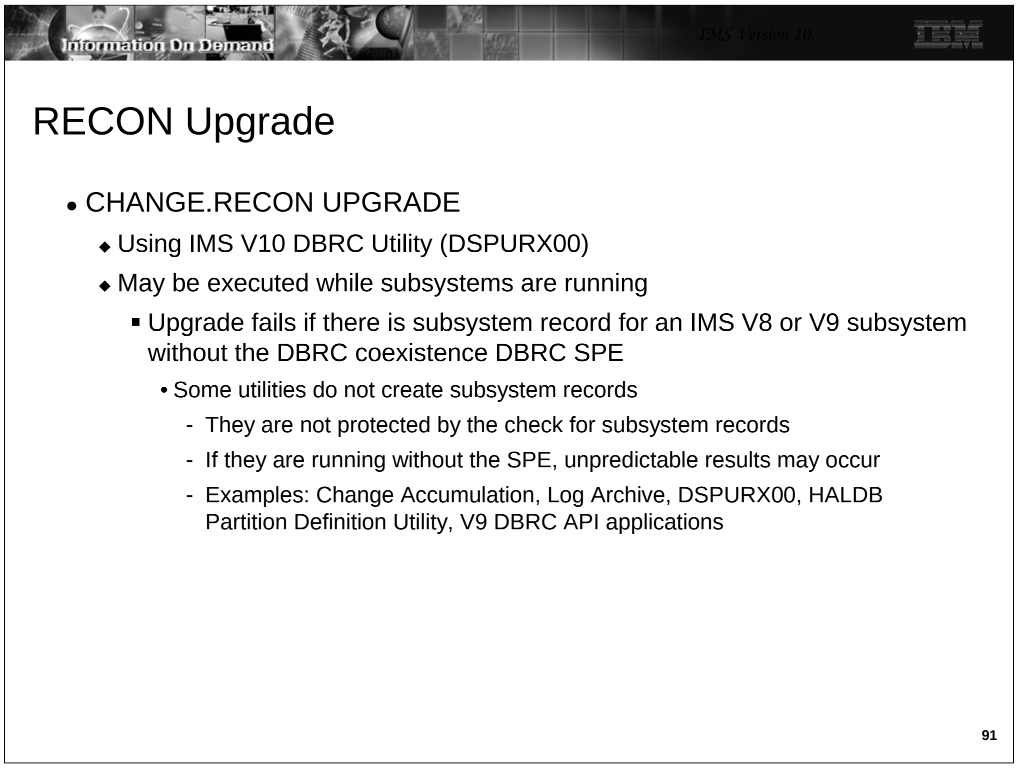
90

IMS V8 RECONs may be upgraded directly to IMS V10. Similarly, IMS V9 RECONs may be upgraded to IMS V10. There is no support to upgrade RECONs from previous releases directly to IMS V10.

PK06145 is an IMS V8 SPE (Small Programming Enhancement) APAR. It allows IMS V8 to use RECONs which have been upgraded to IMS V10.

PK06147 is an IMS V9 SPE APAR. It allows IMS V9 to use RECONs which have been upgraded to IMS V10.

These APARs should be applied to IMS V8 or IMS V9 before its RECONs are upgraded to IMS V10.



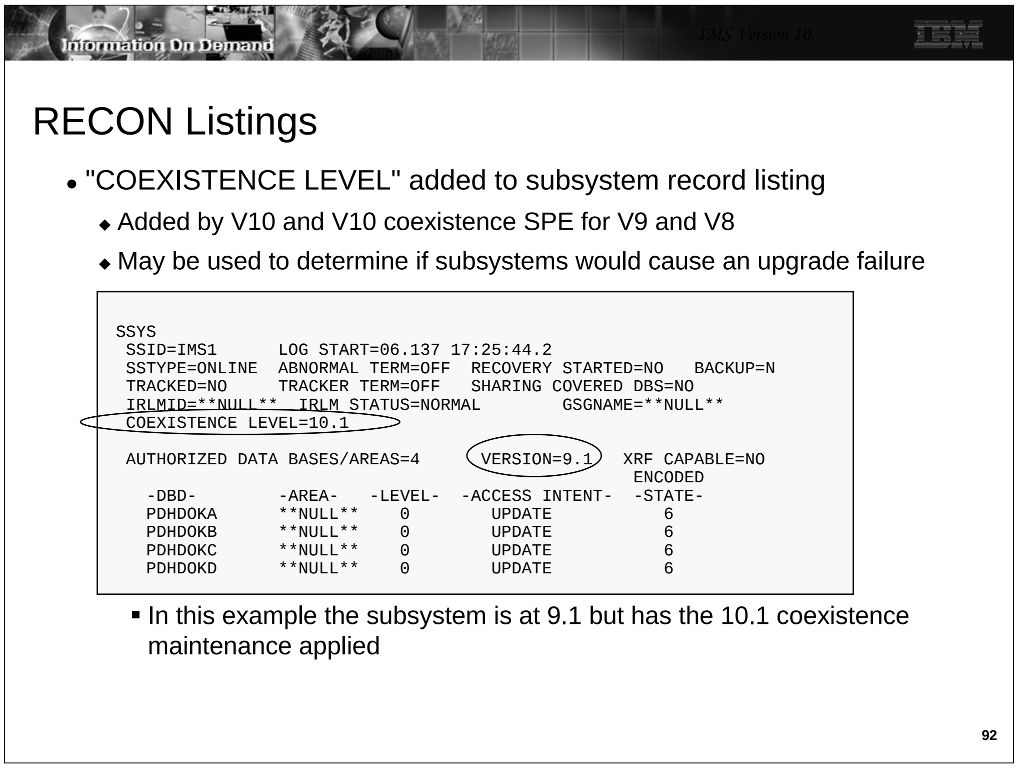
RECON Upgrade

- CHANGE.RECON UPGRADE
  - Using IMS V10 DBRC Utility (DSPURX00)
  - May be executed while subsystems are running
    - Upgrade fails if there is subsystem record for an IMS V8 or V9 subsystem without the DBRC coexistence DBRC SPE
      - Some utilities do not create subsystem records
        - They are not protected by the check for subsystem records
        - If they are running without the SPE, unpredictable results may occur
        - Examples: Change Accumulation, Log Archive, DSPURX00, HALDB Partition Definition Utility, V9 DBRC API applications

91

RECONs are upgraded to IMS V10 by using the DBRC CHANGE.RECON UPGRADE command with the IMS V10 DBRC utility (DSPURX00).

The upgrade may be run while the RECONs are allocated to and being used by IMS V8 or IMS V9, Of course, these systems must be able to use IMS V10 RECONs. The upgrade checks the RECONs to ensure that any subsystems using the RECONs are capable of using IMS V10 RECONs. It does this by examining the SUBSYS records in the RECONs. Some IMS utilities do not create SUBSYS records. Thus, the upgrade cannot determine if they are running. Users must ensure that any IMS utility which is running at the time of the upgrade has the appropriate maintenance (PK06145 or PK06147) which allows it to read IMS V10 RECONs.



RECON Listings

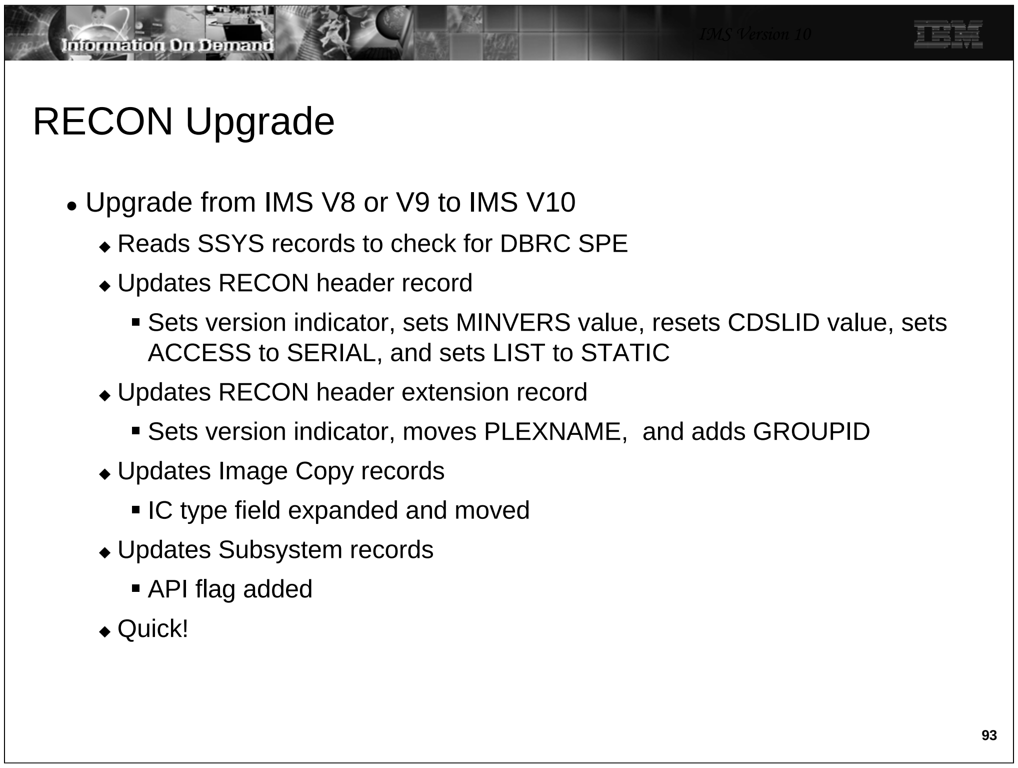
- "COEXISTENCE LEVEL" added to subsystem record listing
  - Added by V10 and V10 coexistence SPE for V9 and V8
  - May be used to determine if subsystems would cause an upgrade failure

```
SSYS
 SSID=IMS1      LOG START=06.137 17:25:44.2
 SSTYPE=ONLINE  ABNORMAL TERM=OFF  RECOVERY STARTED=NO   BACKUP=N
 TRACKED=NO     TRACKER TERM=OFF   SHARING COVERED DBS=NO
 IRLMID=**NULL**  IRLM STATUS=NORMAL        GSGNAME=**NULL**
 COEXISTENCE LEVEL=10.1

 AUTHORIZED DATA BASES/AREAS=4      VERSION=9.1   XRF CAPABLE=NO
                                                   ENCODED
   -DBD-         -AREA-   -LEVEL-  -ACCESS INTENT-  -STATE-
   PDHDOKA      **NULL**    0        UPDATE           6
   PDHDOKB      **NULL**    0        UPDATE           6
   PDHDOKC      **NULL**    0        UPDATE           6
   PDHDOKD      **NULL**    0        UPDATE           6
```

- In this example the subsystem is at 9.1 but has the 10.1 coexistence maintenance applied

92

IMS V10 has added the coexistence level to the RECON listing of subsystem records. This has also been added to IMS V8 and V9 by the IMS V10 coexistence SPEs for these releases. The VERSION= field has existed in previous releases. It indicates the IMS release level of the subsystem. The COEXISTENCE LEVEL= field indicates if the coexistence maintenance for a later release has been applied. In this example, the IMS V10 DBRC coexistence maintenance has been applied to the IMS V9 system used by this subsystem. This listing could have been produced by an IMS V9 DBRC utility with the IMS V10 coexistence SPE applied or it could have been produced by the IMS V10 DBRC utility.



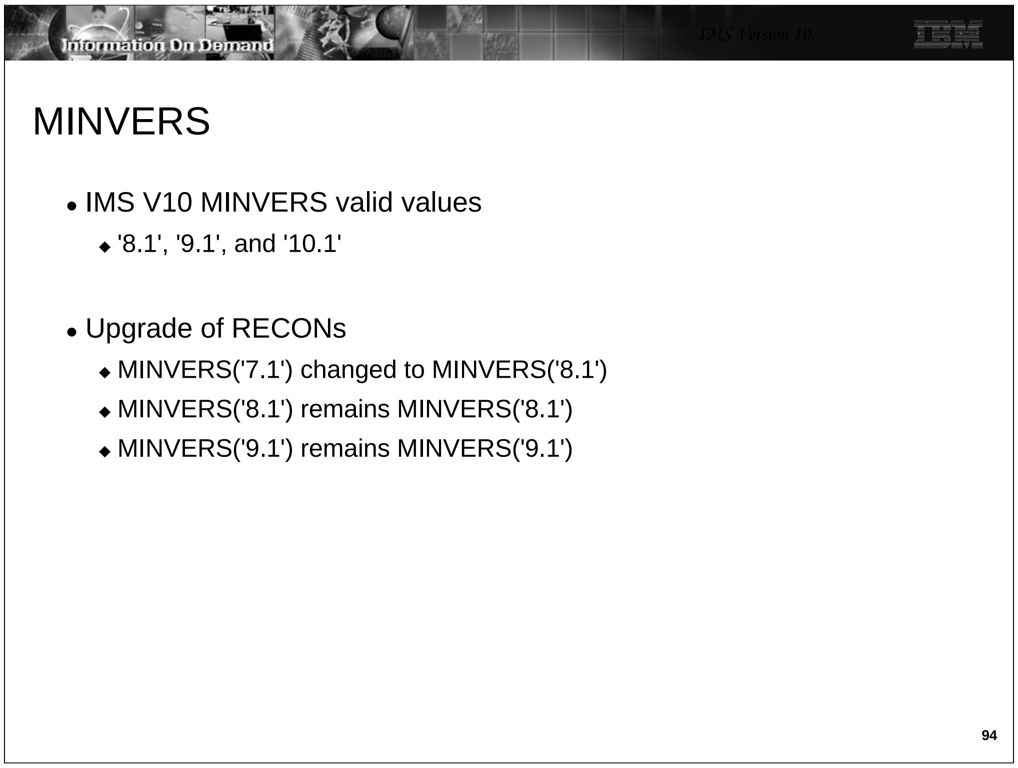
RECON Upgrade

- Upgrade from IMS V8 or V9 to IMS V10
  - Reads SSYS records to check for DBRC SPE
  - Updates RECON header record
    - Sets version indicator, sets MINVERS value, resets CDSLID value, sets ACCESS to SERIAL, and sets LIST to STATIC
  - Updates RECON header extension record
    - Sets version indicator, moves PLEXNAME, and adds GROUPID
  - Updates Image Copy records
    - IC type field expanded and moved
  - Updates Subsystem records
    - API flag added
  - Quick!

The upgrade of the RECONs includes the reading of the subsystem (SSYS) records to ensure that these subsystems are running with the DBRC coexistence SPE. If not, the subsystem could not use the RECONs and the upgrade fails.

The upgrade changes a few records in the RECONs. The header and header extension records are changed to include the correct version indicator and to values for the parameters shown. The Cross DBRC Service Level ID (CDSLID) is set to the higher of the value in the RECONs before the upgrade and "1". The Image Copy records are rewritten to accommodate the larger IC type field. This field was expanded to include new image copy types for FlashCopy and fuzzy user image copies. The subsystem records are rewritten to accommodate the DBRC API flag for subsystems using the DBRC API.

Since only a few records are updated, a typical upgrade will be quick.



Information On Demand

IMS Version 10

MINVERS

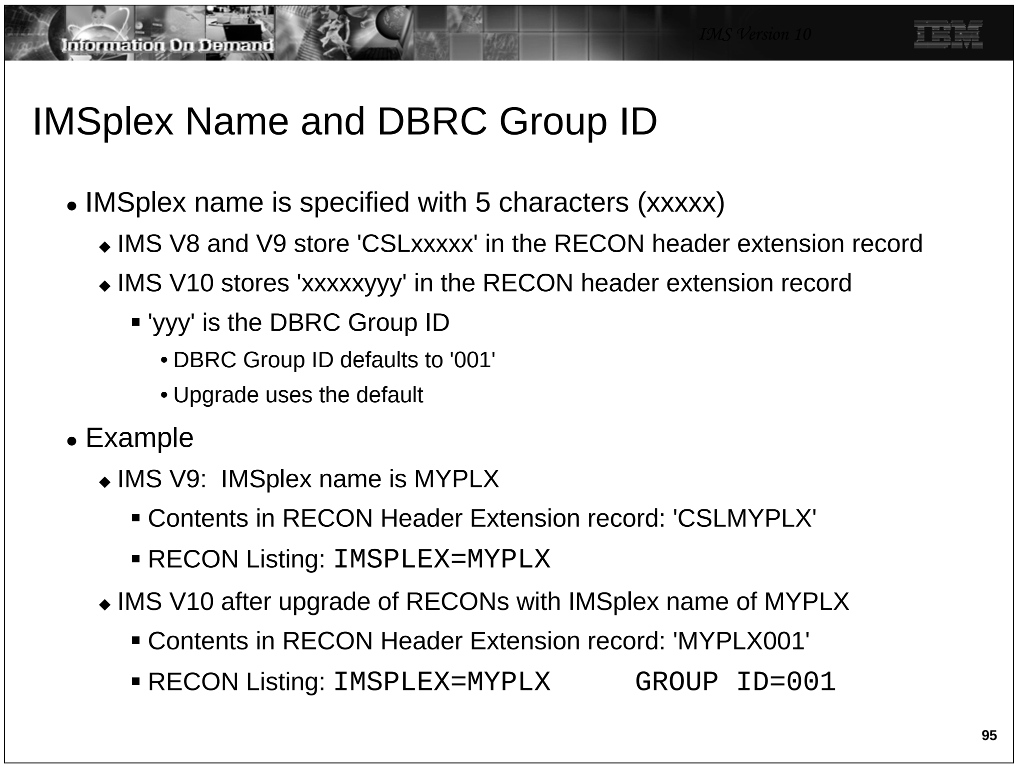
- IMS V10 MINVERS valid values
  - '8.1', '9.1', and '10.1'

- Upgrade of RECONs
  - MINVERS('7.1') changed to MINVERS('8.1')
  - MINVERS('8.1') remains MINVERS('8.1')
  - MINVERS('9.1') remains MINVERS('9.1')

94

MINVERS is the parameter on the INIT.RECON and CHANGE.RECON commands which controls the minimum level of IMS which may use the RECONs. The minimum level of IMS which can use V10 RECONs is V8. If the previous MINVERS value was for '7.1', it is changed to '8.1' by the upgrade. Otherwise, upgrades do not change the MINVERS value.

MINVERS values of 81 and 91 (specified without the decimal point or the single quotes) are accepted by IMS V10 for compatibility, however, using the single quotes and the decimal point are required for specifying V10 ('10.1') and recommended for specifying V8 ('8.1') and V9 ('9.1').

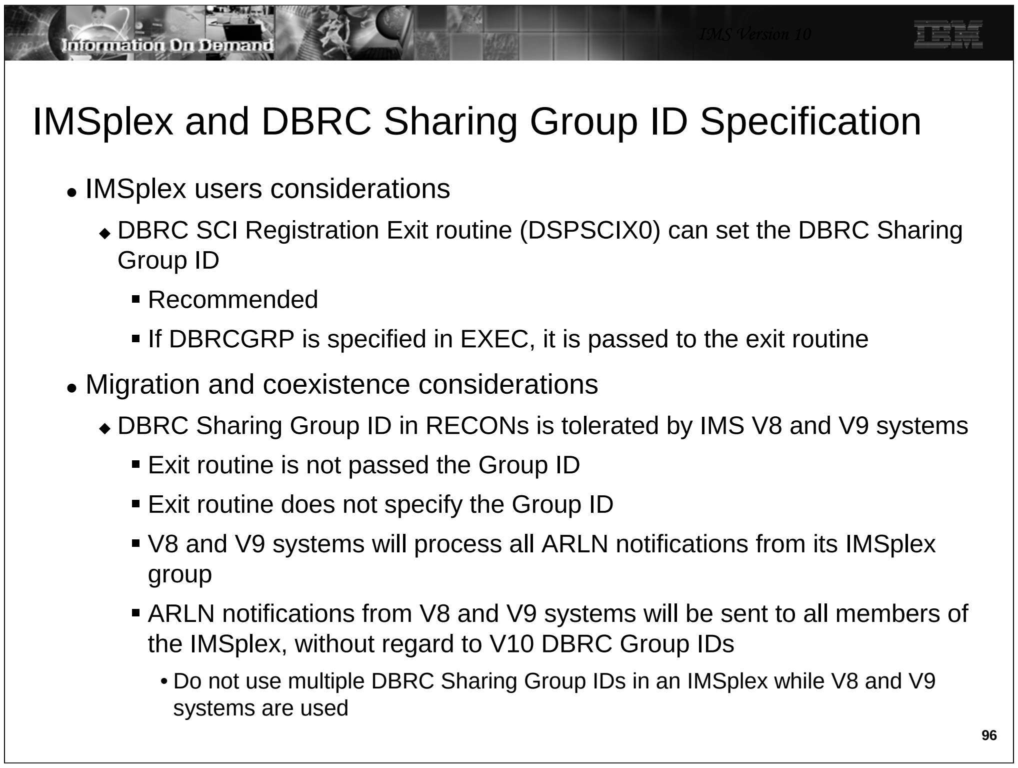


IMSplex Name and DBRC Group ID

- IMSplex name is specified with 5 characters (xxxxx)
  - IMS V8 and V9 store 'CSLxxxxx' in the RECON header extension record
  - IMS V10 stores 'xxxxxyyy' in the RECON header extension record
    - 'yyy' is the DBRC Group ID
      - DBRC Group ID defaults to '001'
      - Upgrade uses the default
- Example
  - IMS V9: IMSplex name is MYPLX
    - Contents in RECON Header Extension record: 'CSLMYPLX'
    - RECON Listing: `IMSPLEX=MYPLX`
  - IMS V10 after upgrade of RECONs with IMSplex name of MYPLX
    - Contents in RECON Header Extension record: 'MYPLX001'
    - RECON Listing: `IMSPLEX=MYPLX      GROUP ID=001`

The IMSplex name is optional. It is required for Automatic RECON Loss Notification and Parallel RECON Access. The IMSplex name is specified with up to 5 characters. It is specified either in the IMSPLEX= execution parameter or by the DBRC SCI Registration exit routine. When first specified, it is stored in the RECONs. IMS V8 and V9 store the IMSplex name as 'CSLxxxxx' where 'xxxxx' is the value specified in the IMSPLEX= parameter or in the exit routine. When the RECONs are upgraded to V10, the value stored is 'xxxxxyyy' where 'yyy' is the DBRC Group ID. The upgrade sets the DBRC Group ID to '001' which is the default value.

IMS V8, V9, and V10 list only the 5 characters of the IMSplex name in listings of the RECON header. These listing include a line with IMSPLEX=xxxxx when an IMSplex name has been stored in the RECONs. If there is no value stored, the line includes IMSPLEX=**NONE**. IMS V10 listings also include the DBRC Group ID on this line. If there is no IMSplex name the Group ID is listed as GROUP=**NONE**. If there is an IMSplex name, the Group ID is listed as GROUP=yyy where yyy is the Group ID.



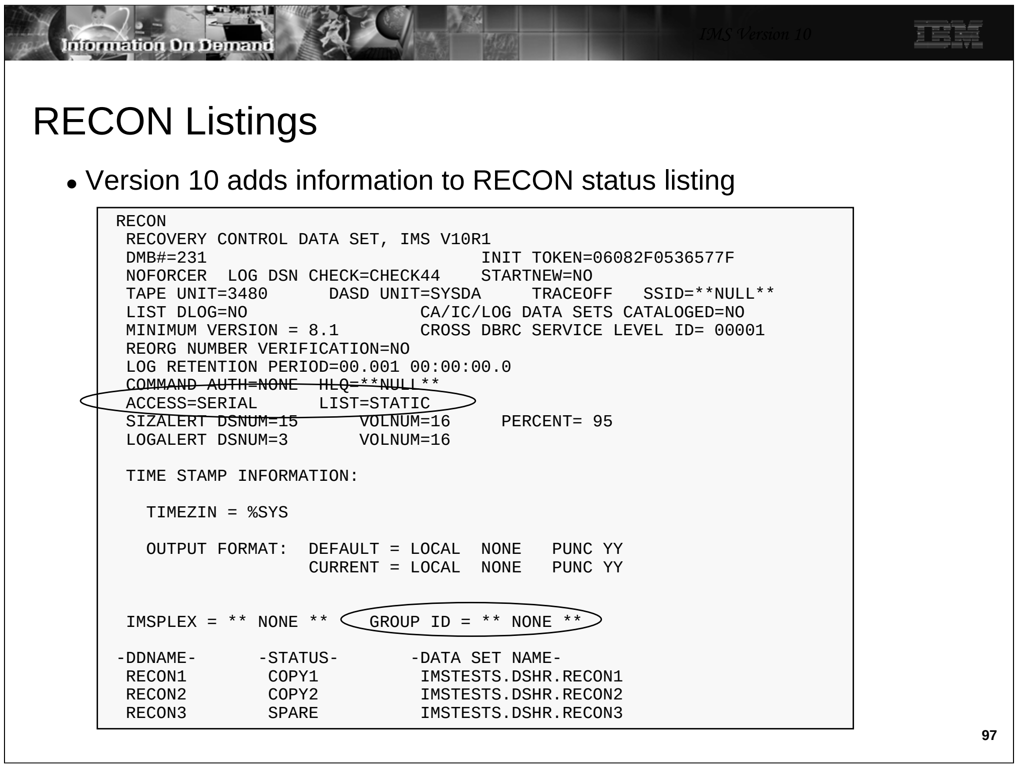
IMSplex and DBRC Sharing Group ID Specification

- IMSplex users considerations
  - DBRC SCI Registration Exit routine (DSPSCIX0) can set the DBRC Sharing Group ID
    - Recommended
    - If DBRCGRP is specified in EXEC, it is passed to the exit routine
- Migration and coexistence considerations
  - DBRC Sharing Group ID in RECONs is tolerated by IMS V8 and V9 systems
    - Exit routine is not passed the Group ID
    - Exit routine does not specify the Group ID
    - V8 and V9 systems will process all ARLN notifications from its IMSplex group
    - ARLN notifications from V8 and V9 systems will be sent to all members of the IMSplex, without regard to V10 DBRC Group IDs
      - Do not use multiple DBRC Sharing Group IDs in an IMSplex while V8 and V9 systems are used

96

The DBRC SCI Registration exit routine (DSPSCIX0) may be used to specify the IMSplex name, as in previous releases, and the new DBRC Group ID. The use of the exit routine is recommended for users of IMSplex. It removes the requirement to specify IMSPLEX= for the execution of all IMS jobs which use DBRC. This includes batch jobs and utilities. With IMS V10 the exit also may specify the DBRC Sharing Group ID. This removes the requirement to specify DBRCGRP= for IMS executions.

IMS V8 and V9 systems can tolerate the specification of the DBRC Group ID in the RECONs. DBRCGRP= is not a valid parameter on the EXEC statement for V8 and V9. When the exit routine is invoked in a V8 or V9 environment, the DBRC Group ID is not passed to it. The exit routine cannot specify the DBRC Group ID. Even though a V8 or V9 instance cannot specify the DBRC Group ID, it can join an IMSplex where V10 instances are using DBRC Group IDs. The V8 or V9 instance will be passed all ARLN notifications from the IMSplex group. If a V8 or V9 system reconfigures its RECONs, its ARLN notification will be processed by all members of the IMSplex. This will include all V10 systems. If there are multiple DBRC Groups, all members of all groups will process the notification. For these reasons, you should not use multiple DBRC Group IDs in an IMSplex while you are still using IMS V8 or V9 systems.



IMS Version 10

RECON Listings

- Version 10 adds information to RECON status listing

```
RECON
 RECOVERY CONTROL DATA SET, IMS V10R1
 DMB#=231                           INIT TOKEN=06082F0536577F
 NOFORCER  LOG DSN CHECK=CHECK44    STARTNEW=NO
 TAPE UNIT=3480      DASD UNIT=SYSDA     TRACEOFF   SSID=**NULL**
 LIST DLOG=NO                 CA/IC/LOG DATA SETS CATALOGED=NO
 MINIMUM VERSION = 8.1        CROSS DBRC SERVICE LEVEL ID= 00001
 REORG NUMBER VERIFICATION=NO
 LOG RETENTION PERIOD=00.001 00:00:00.0
 COMMAND AUTH=NONE  HLQ=**NULL**
 ACCESS=SERIAL      LIST=STATIC
 SIZALERT DSNUM=15        VOLNUM=16     PERCENT= 95
 LOGALERT DSNUM=3         VOLNUM=16

 TIME STAMP INFORMATION:

   TIMEZIN = %SYS

   OUTPUT FORMAT:  DEFAULT = LOCAL  NONE   PUNC YY
                   CURRENT = LOCAL  NONE   PUNC YY


 IMSPLEX = ** NONE **    GROUP ID = ** NONE **

-DDNAME-      -STATUS-       -DATA SET NAME-
 RECON1        COPY1          IMSTESTS.DSHR.RECON1
 RECON2        COPY2          IMSTESTS.DSHR.RECON2
 RECON3        SPARE          IMSTESTS.DSHR.RECON3
```

97

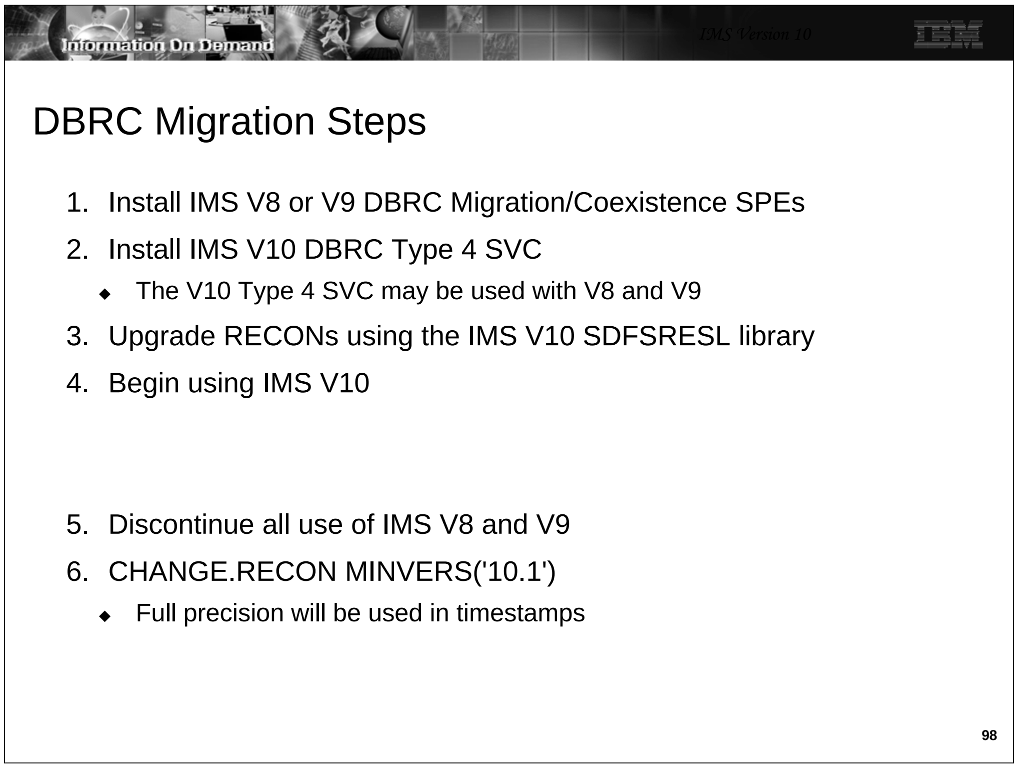
The RECON status or header listing has some added and changed information.

Of course, the IMS version is listed as "V10R1". This means that the RECONs have been upgraded to IMS V10.

There is a new line which lists the type of RECON access, either SERIAL or PARALLEL. On the same line the default for the DBRC LIST command, either STATIC or CONCURR, is shown.

On the line where the IMSPLEX value is shown, the DBRC Group ID value is also shown. In this example, these parameters have no values so "** NONE **" is listed.

The sample listing shown here includes the "CROSS DBRC SERVICE LEVEL ID". This also appears on IMS V9 RECON listings when the maintenance for APARs PQ98655 and PK01097 is applied and on IMS V8 RECON listings when the maintenance for APARs PQ98654 and PK01096 is applied. The service level ID is used to invoke functions which require a consistent level of maintenance on all IMS systems using the RECONs.



IMS Version 10

DBRC Migration Steps

1. Install IMS V8 or V9 DBRC Migration/Coexistence SPEs
2. Install IMS V10 DBRC Type 4 SVC
   - The V10 Type 4 SVC may be used with V8 and V9
3. Upgrade RECONs using the IMS V10 SDFSRESL library
4. Begin using IMS V10

5. Discontinue all use of IMS V8 and V9
6. CHANGE.RECON MINVERS('10.1')
   - Full precision will be used in timestamps

98

This shows the DBRC steps for migration to IMS V10.

The first set of steps allows you to begin using IMS V10. The migration/coexistence SPE must be installed on the old release before you upgrade the RECONs to V10. The V10 DBRC Type 4 SVC must be installed before you may use IMS V10. The upgrade of the RECONs to IMS V10 requires that you use the SDFSRESL library created by the installation of IMS V10. The upgrade using this library will be to the IMS V10 format. Once the RECONs have been upgraded, you may begin using IMS V10. You may also continue to use IMS V8 or V9.

Once you have discontinued all use of IMS V8 and V9, you can change the MINVERS value to '10.1'. This causes IMS to begin using the increased precision timestamp. Before changing MINVERS to '10.1', you must ensure that the IMS utility control statements that you use specify full precision in their timestamps. The control statements generated by GENJCL statements will always generate control statements with the correct timestamps. Remember that the position of the timestamp in the control statements for the IMS V10 Change Accumulation and Database Recovery utilities does not depend on the MINVERS value, however, if MINVERS is not '10.1' the low order part of the timestamp does not matter since these positions in timestamps are not recorded in the RECONs unless MINVERS('10.1') is specified.