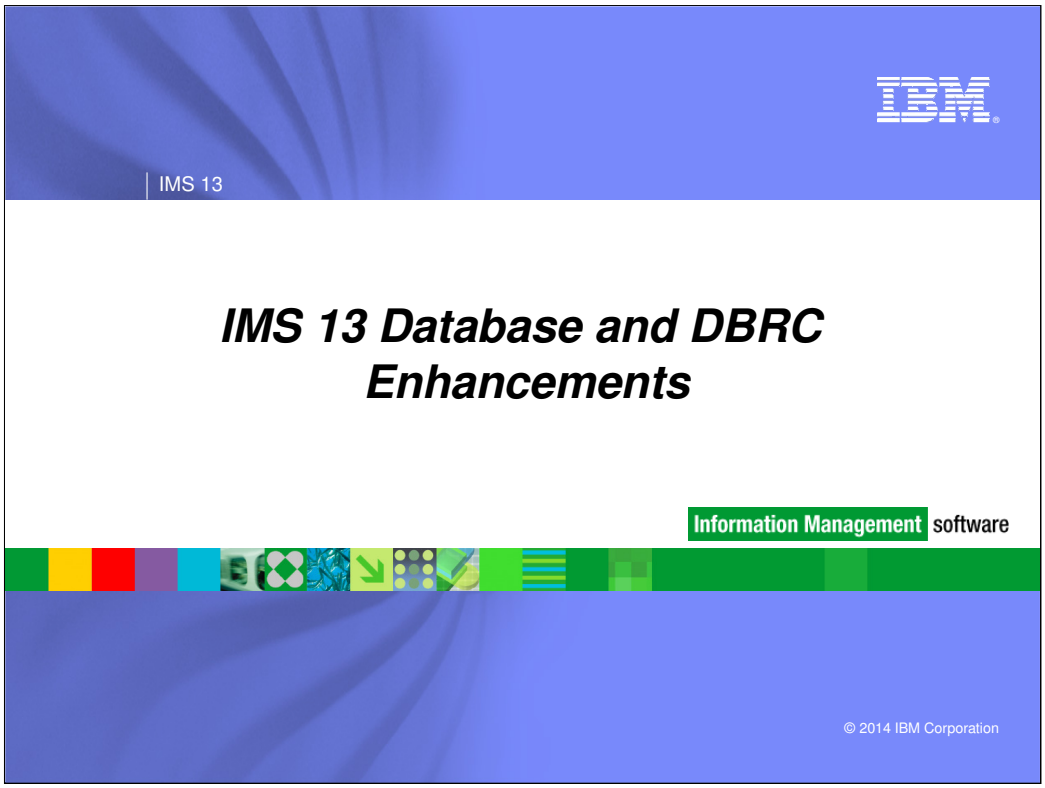

IBM
IMS 13
IMS 13 Database and DBRC Enhancements
Information Management software
© 2014 IBM Corporation

Database and DBRC Enhancements

- Database Versioning
- HALDB Alter
- Fast Path Enhancements
  - DEDB Alter
  - Secondary Index Enhancements
- DBRC Migration and Coexistence
- DELETE.LOG INACTIVE and TOTIME (KFN0547)

08- IMS 13 DB & DBRC: 504

Database Versioning

08- IMS 13 DB & DBRC: 505

Database Versioning

- IMS 13 allows application programs to use different versions of the same physical database
  - Multiple views of the physical data are maintained in the IMS catalog
  - Application programs can use different views of the same physical IMS database
- Benefit
  - Customers can support multiple versions of an IMS database
  - Physical database structure can be changed without having to modify all the existing application programs using the database

Database Versioning Support is for Full Function, HALDB and DEDB database customers who need support for multiple views of the physical data to a variety of application needs such as:

•Implementing application changes over time.

•Ability to use application programs, for which there is no source code, after database structure changes.

Versioning support enables users to assign user-defined version IDs to different versions of the structure of a database. The user-defined version IDs are stored in the record for the database in the IMS Catalog. Upon accessing the database, application programs specify the version of the database that they need. If they do not specify a version, by default they will get the version of the database structure at the current level.

Database Versioning Prerequisites

- Software requirements
  - Same as IMS Version 13
  - IMS Catalog
  - DBRC RECON MINVERS (‘13.1’)
- Hardware requirements
  - Same as IMS Version 13

08- IMS 13 DB & DBRC: 507

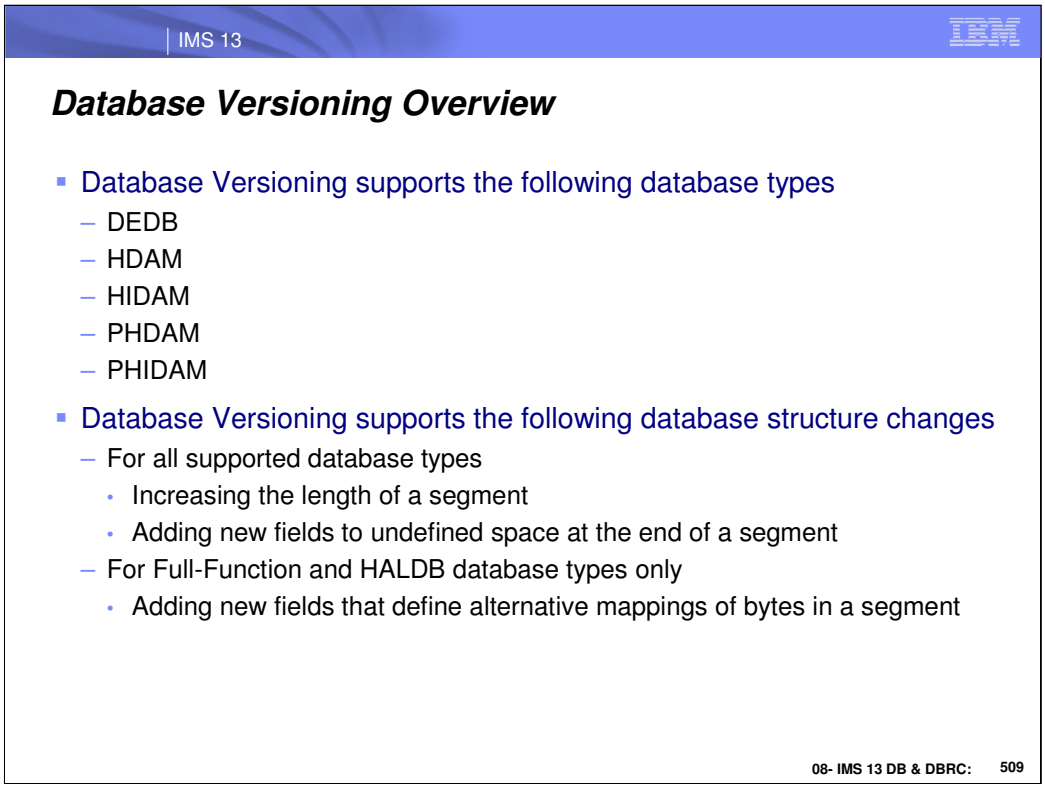
Database Versioning Overview

- Provides the ability to assign user-defined version identifiers to different versions of an IMS database structure
- Enables structural changes to a database while providing multiple views of the physical IMS data to various applications
- Applications referencing a new physical database structure can be brought online without affecting applications that use previous database structures
- Applications not requiring sensitivity to the new physical database structure can continue to access the database without any modifications or recompilation

Can be used in conjunction with IMS 13 Database ALTER function.

Database versioning provides the ability to assign user-defined version identifiers to different versions of the structure of a database. These identifiers enable you to make structural changes to a database while providing multiple views of physical data to a variety applications.

New applications that reference a newer structure of a database can be brought online without affecting applications that use previous database structures. Unchanged applications, which do not have to be sensitive to the new physical structure of the database, can continue to access the database.



Database versioning can be used for the following types of databases:

- DEDB
- HDAM
- HIDAM
- PHDAM
- PHIDAM

IMS database versioning supports databases that have logical relationships and databases that have secondary indexes.

The database versioning function:

Can be used in conjunction with the database alter functions to keep track of different versions of the structure of a database. Supports the following structural changes to all supported database types:

- Increasing the length of a segment.
- Adding a new field at the end of a segment

  Supports the following structural changes to FF and HALDB database types
- Adding a new field to a segment that defines an alternative mapping of bytes in segment.

Database Versioning only supports changes which include increasing the length of the segment and defining new fields. These changes are normally implemented by recoding the DBD source and running the DBD, PSB, and ACBGEN utilities. The customer will then unload/reload the database or utilize the IMS 13 HALDB Alter function followed by performing an Online Change (OLC).

Changes made to any existing fields, which include changing the starting position or length of the field are not allowed.

Database Versioning Overview

- Before Database Versioning is established for a database
  - IMS continues to only recognize the current physical database definition
- Version numbers must be maintained in incremental values
  - Specified on the DBD
- Applications programs can specify a desired database version
  - System default setting
  - PSB setting
  - PCB in the PSB
  - DL/I INIT call

Requirements:

To enable the database versioning support, the following tasks are required:

•Specify the new parameter, DBVERSION=Y, in the databases section in the DFSDFxxx member of the IMS PROCLIB data set to indicate that database versioning is to be used.

•If Database Versioning is enabled, the IMS Catalog is required to be available in order to retrieve the correct DBD version for the application programs. If the IMS Catalog is not available then the application is returned an 'NA' status code.

•Application programs that need to access a particular version of a database definition can specify the DBVER= on the PCB statement of the PSB source, or issue the INIT VERSION call to specify the database version for each database view that is used by the application.

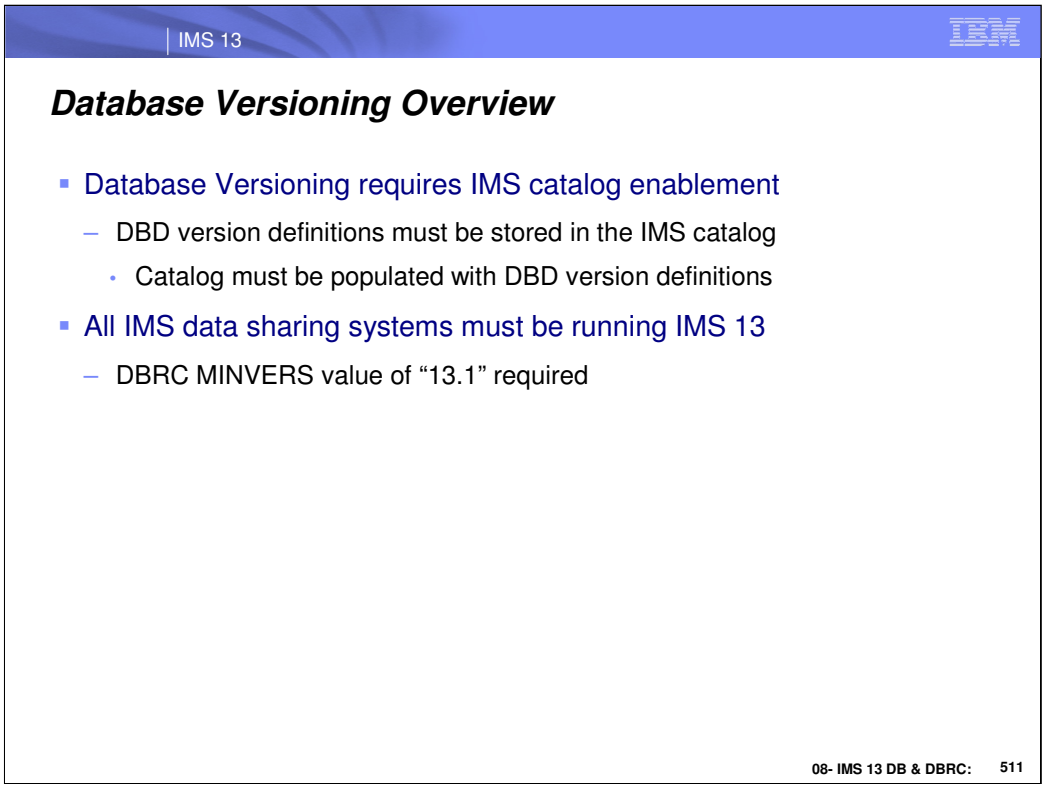
When a version number is assigned to the DBD, if the database is logically related to one or more databases, all logically-related databases must be included in the ACBGEN process.

The affected database descriptions (DBDs) must be in the IMS catalog. The IMS catalog metadata describes the current and previous structures of a database. The metadata includes the version numbers that identify each structure of a database. When an application program makes a call to a versioned database, IMS internally references the catalog to determine which structure corresponds to the provided version number and whether the format of the requested data needs to be modified before it is returned to the application program.

After a DBD has a version number, apps can use the new INIT VERSION call to access a specific version of the database or specify the database version for an application program on the DBVER parm in the PCB statement. The following requirements apply:

•INIT VERSION call must be issued before issuing a DL/I call to access that database

•Version specified on INIT VERSION call overrides version number specified on the PCB



Requirements:

•Once Database Versioning is enabled, the IMS Catalog is required to be available in order to retrieve the correct DBD version for the application programs. If the IMS Catalog is not available then the application is returned an ‘NA’ status code. The IMS catalog must be enabled!

•The affected database descriptions (DBDs) must be in the IMS catalog.

•After a change is implemented and a version number is assigned to the DBD, if the database is logically related to one or more databases, all logically-related databases must be included in the ACBGEN process.

The IMS catalog metadata describes the current and previous structures of a database. The metadata includes the version numbers that identify each structure of a database. When an application program makes a call to a versioned database, IMS internally references the catalog to determine which structure corresponds to the provided version number and whether the format of the requested data needs to be modified before it is returned to the application program.

After a DBD has a version number, apps can use the new INIT VERSION DL/I call to access a specific version of the database or specify the database version for an application program on the DBVER parm in the PCB statement. The following requirements apply:

•INIT VERSION call must be issued before using a DL/I call to access that database.

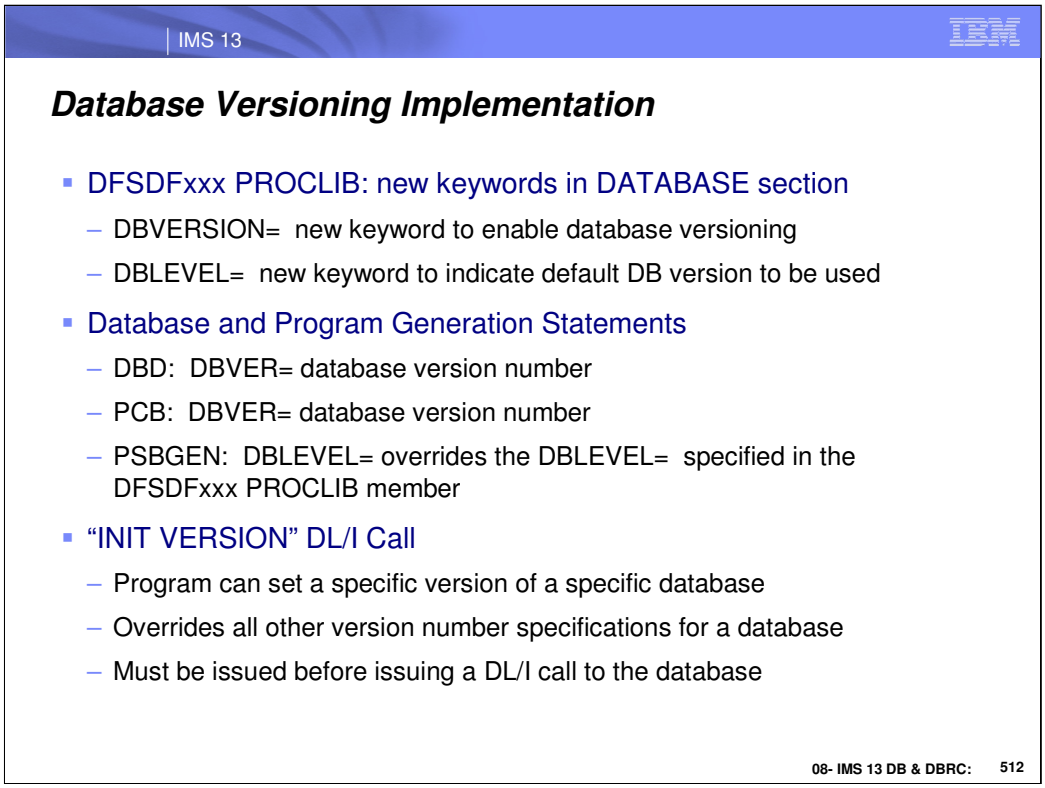
•Version specified on INIT VERSION call overrides version number that is specified on the PCB statement.

To enable the database versioning support, the following tasks are required:

•Specify the new parameter, DBVERSION=Y, in the databases section in the DFSDFxxx member of the IMS PROCLIB data set to indicate that database versioning is to be used.

•IMS Catalog is required and needs to be setup.

•Application program that needs to access a particular version of a database definition can specify the DBVER= on the PCB statement of the PSB source, or issue the INIT VERSION call to specify the database version for each database view that is used by the application.



The following IMS™ components are updated to support database versioning:

DFSDFxxx PROCLIB member

Two new keywords are added to the DATABASE section of the DFSDFxxx PROCLIB member that specify whether database versioning is enabled and, if so, what default versioning is to be used:

DBVERSION=

DBLEVEL=

DBD and PSB generation statements

The DBD and PCB statements are enhanced with a new DBVER parameter, where you can specify the database version number:

DBVER=

The PSBGEN statement is enhanced with a new parameter that can be used to override the default versioning that is specified in the DFSDFxxx PROCLIB member:

DBLEVEL

After a DBD has a version number, apps can use the new INIT VERSION DL/I call to access a specific version of the database or specify the database version for an application program on the DBVER parm in the PCB statement. The following requirements apply:

- **Version specified on INIT VERSION call overrides version number specified on the PCB statement.**
- **INIT VERSION call must be issued before using a DL/I call to access that database.**

Implementation – DFSDFxxx PROCLIB Member

- New keywords added to DFSDFxxx DATABASE section
  - DBVERSION = Y | N
    - Enables database versioning
    - Database versioning is disabled by default
  - DBLEVEL = CURR | BASE
    - Ignored when DBVERSION=N
    - CURR (default)
      - IMS returns data from all databases using the current DBD version, which is the current physical level, unless a specific database version is requested
    - BASE
      - IMS returns data from all databases using the lowest DBD version number retrieved from the IMS Catalog, unless a specific database version is requested

DBVERSION = Y|N

Specifies that database versioning is to be enabled. When parm omitted, database versioning is disabled.

When database versioning is enabled, IMS becomes sensitive to the version number associated with a DBD. Applications may then request a specific DBD version by specifying the version number on the PCB DBVER= parameter. If the requested DBD version is not the currently active one, then IMS will retrieve it from the IMS catalog. If the requested DBD version does not exist, IMS returns a bad status code to the application.

When database versioning is not enabled, IMS is not sensitive to the version number associated with a DBD and application programs continue to access the data from the database at the physical database level. This is how IMS operates prior to V13. If an application program specifies a DBD version on the PCB DBVER= parameter while database versioning is not enabled, IMS returns a bad status code to the application.

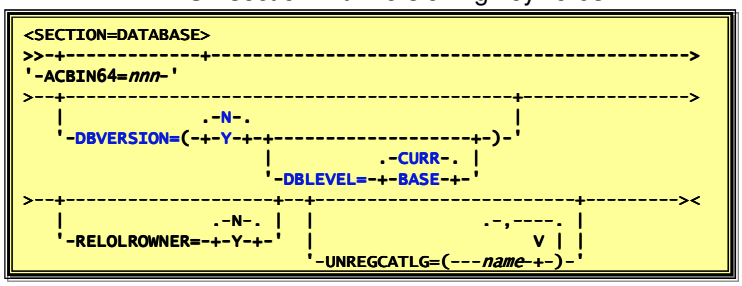
DBLEVEL=CURR|BASE: Specifies the default database version level. It is only used if DBVERSION=Y, otherwise it is ignored. The default setting is CURR.

If DBLEVEL=CURR, then for all application programs IMS will return data from the database at the current physical level unless the application requested a specific DBD version on the PCB DBVER= parameter. DBLEVEL=CURR is recommended. It allows for a seamless transition over to database versioning and allows application development to move forward as long as all application programs are maintained to access databases at the latest level. For any application program that requires the continued use of older versions of a DBD, the PCB DBVER= parameter may be used.

If DBLEVEL=BASE, then for all application programs IMS will return data from the database at the original base level (ie., version 0) using the database definition stored in the IMS catalog. This is unless the application requests a specific DBD version on the PCB DBVER= parameter. DBLEVEL=BASE should only be considered if you have a large number of application programs that can not be changed (ie. pgm source or PSB no longer available). This allows application programs to continue using the original database definition, while also allowing new application development & new database structural changes. The original database definition is retrieved from the IMS catalog. New applications or applications that expect data returned at the latest database level must specify the PCB DBVER= parameter or DBLEVEL=CURR on the PSBGEN statement.

Implementation – DFSDFxxx PROCLIB Member

DFSDFxxx DATABASE section with versioning keywords



SAMPLE DFSDFxxx DATABASE section with versioning support enabled

```
/*********************************************************************/
/* Database Section                                                  */
/*********************************************************************/
<SECTION=DATABASE>
ACBIN64=8                          /* Create 64-bit storage pool */
DBVERSION=(Y,DBLEVEL=BASE)           /* Change Default to BASE */
RELOLROWNER=Y     /* Release ownership of OLR when IMS terminates */
/*********************************************************************/
/*                                                                   */
/*********************************************************************/
```


DBVERSION = Y|N

Specifies that database versioning is to be enabled. When parm omitted, database versioning is disabled.

When database versioning is enabled, IMS becomes sensitive to the version number associated with a DBD. Applications may then request a specific DBD version by specifying the version number on the PCB DBVER= parameter. If the requested DBD version is not the currently active one, then IMS will retrieve it from the IMS catalog. If the requested DBD version does not exist, IMS returns a bad status code to the application.

When database versioning is not enabled, IMS is not sensitive to the version number associated with a DBD and application programs continue to access the data from the database at the physical database level. This is how IMS operates prior to V13. If an application program specifies a DBD version on the PCB DBVER= parameter while database versioning is not enabled, IMS returns a bad status code to the application.

DBLEVEL=CURR|BASE: Specifies the default database version level. It is only used if DBVERSION=Y, otherwise it is ignored. The default setting is CURR.

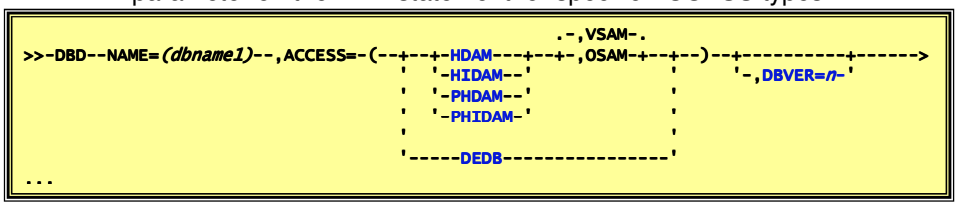
If DBLEVEL=CURR, then for all application programs IMS will return data from the database at the current physical level unless the application requested a specific DBD version on the PCB DBVER= parameter. DBLEVEL=CURR is recommended. It allows for a seamless transition over to database versioning and allows application development to move forward as long as all application programs are maintained to access databases at the latest level. For any application program that requires the continued use of older versions of a DBD, the PCB DBVER= parameter may be used.

If DBLEVEL=BASE, then for all application programs IMS will return data from the database at the original base level (ie., version 0) using the database definition stored in the IMS catalog. This is unless the application requests a specific DBD version on the PCB DBVER= parameter. DBLEVEL=BASE should only be considered if you have a large number of application programs that can not be changed (ie. pgm source or PSB no longer available). This allows application programs to continue using the original database definition, while also allowing new application development & new database structural changes. The original database definition is retrieved from the IMS catalog. New applications or applications that expect data returned at the latest database level must specify the PCB DBVER= parameter or DBLEVEL=CURR on the PSBGEN statement.

Implementation – Database Generation Statement

- New parameter added to DBD Statement
  - DBVER=*n*
    - Specifies a DBD version number to be associated with a database structure change
    - Supports ACCESS types DEDB, HDAM, HIDAM, PHDAM & PHIDAM
    - Numeric values from 1 – 2147483647 (*2 Gigs*)

DBVER= parameter on the DBD statement for specific ACCESS types



Database DBD statement

A new parameter, DBVER=, is added to the DBD statement. The new parameter allows the user to specify a version number associated with a structure change being made to the DBD. The DBVER parameter is only supported for DEDB, HDAM, HIDAM, PHDAM, and PHIDAM. Version numbers specified on this parameter must be maintained in incremental values.

Specifies the version number used to associate a change made to the DBD. It has to be a numeric value, and the valid value range is 1 – 2147483647.

Implementation – PCB Statement in PSB

- New parameter added to PCB Statement
  - DBVER=*n*
    - Specifies the version of the DBD to use when accessing the database
    - Must match a defined DBD version number stored in the IMS catalog
    - If multiple PCBs within a PSB refer to the same database, each PCB must specify the same DBD version number
    - Numeric values from 0 – 2147483647 (*2 Gigs*)
    - If not specified, the DBD version used depends on DBLEVEL= parm in the PSBGEN statement or the DFSDFxxx PROCLIB member

DBVER= parameter on the PCB statement

```
>>-+-----------+--PCB--TYPE=DB--+-,DBDNAME=-+--name----------------+----------+------>
   '-label-----'                '-,NAME=----'                       '-,DBVER=n-'

...
```


Program PCB Statement

A new parameter, DBVER=, is added to specify the version of the DBD to be used when accessing the associated database. The valid values are between 0 – 2147483647.

When database versioning is enabled, and the DBVER= parameter is omitted from the program PCB statement, the DBD version used to return data depends on the DBLEVEL= parameter in the PSBGEN statement or the DFSDFxxx PROCLIB member

DBVER=

Specify the version number of the database DBD to access for this application program. It has to be a numeric value, and valid value range is 0 - 2147483647. If multiple PCBs within a PSB refer to the same database, then each PCB must specify the same DBD version number.

DBVER is only supported for TYPE=DB. If specified for TYPE=TP or TYPE=GSAM, it is ignored.

When coding the PSB with multiple references to the same database, if the same version number is not used then PSBGEN will fail with an MNOTE.

Not allowed:

PCB TYPE=DB,DBNAME=DBJK21,DBVER=2

PCB TYPE=DB,DBNAME=DBJK21,DBVER=1

PCB TYPE=DB,DBNAME=DBJK22,DBVER=3

PSBGEN PSBNAME=PSBJK

Allowed:

PCB TYPE=DB,DBNAME=DBJK21,DBVER=2

PCB TYPE=DB,DBNAME=DBJK21,DBVER=2

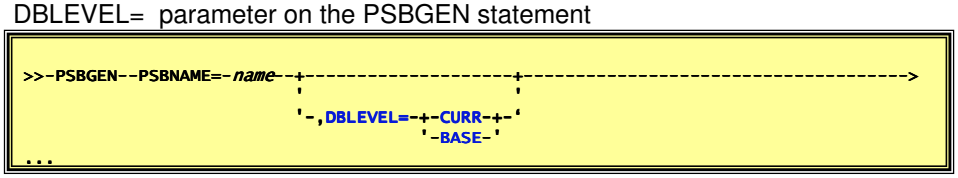
PCB TYPE=DB,DBNAME=DBJK22,DBVER=3

PSBGEN PSBNAME=PSBJK

Implementation – Program Generation Statement

- New parameter added to PSBGEN Statement
  - DBLEVEL=CURR | BASE
    - Specifies the default database version level returned to programs using this PSB and not requesting specific database versions
      - DBLEVEL=CURR
        - PCBs within the PSB will access the database using the current physical structure
      - DBLEVEL=BASE
        - PCBs within the PSB will use the lowest base version in the IMS catalog
    - Overrides the default setting for DBLEVEL specified in DFSDFxxx PROCLIB member

DBLEVEL= parameter on the PSBGEN statement



08- IMS 13 DB & DBRC: 517

PSBGEN statement

DBLEVEL=CURR | BASE

When database versioning is enabled, specifies the default database version level that is returned to application programs that do not request a specific database version. For all application programs that use this PSB, the value specified here overrides the overall default system setting for DBLEVEL if specified in the DFSDFxxx PROCLIB member. Any DB PCB which does not request a specific database version will follow this default rule.

DBLEVEL=BASE

When any PCB within this PSB does not specify a database version on the DBVER parameter, IMS returns data that conforms to the database structure that is defined by the lowest version number in the DBD record of the

database in the IMS catalog.

DBLEVEL=CURR (default)

When any PCB within this PSB does not specify a database version on the DBVER parameter, IMS returns data that conforms to the database structure that is defined by the highest version number. The highest version number defines the actual current structure of the physical database.

If database versioning is disabled, then this parameter is ignored.

Implementation – DL/I INIT Call

- New VERSION function added to the “INIT” call interface
  - Application program can specify the database name(s) and the DBD version(s) to be used when making DL/I calls to the database(s)
  - Specified version number must match a version number defined on a DBD for the named database and stored in the IMS catalog
  - Takes precedence over all other version number specifications and defaults (ie. PCB statement, PSBGEN statement, DFSDFxxx)
  - An INIT call used to version a database has to be executed prior to the first DL/I call for the database, but not before first “GU” call to IOPCB
  - Can only issue one INIT VERSION call for a specific database within an application

When database versioning is enabled, an application program can use the "VERSION" function to request a version of a database that is different from the version number that is specified for the application program on the PCB or from the default version that is returned by IMS. A version number specified on the INIT VERSION call takes precedence over all other version specifications and defaults.

When the INIT VERSION call is not issued prior to a DL/I to access a database, the version of the database that is returned to the application program is determined by the DBVER keyword of the PCB statement. If the DBVER keyword is not specified, IMS returns either returns the highest or lowest version number, as determined by the DBLEVEL keyword in either the PSBGEN statement or the database section of the DFSDFxxx PROCLIB member.

Each database name is specified by using alphabetic characters and can be specified only once. Specify only names of physical databases. The names of logical databases are not supported.

Each version is specified as a numeric value from 0 to 2147483647. The number specified must match a version number defined on a DBD for the named database and stored in the IMS catalog.

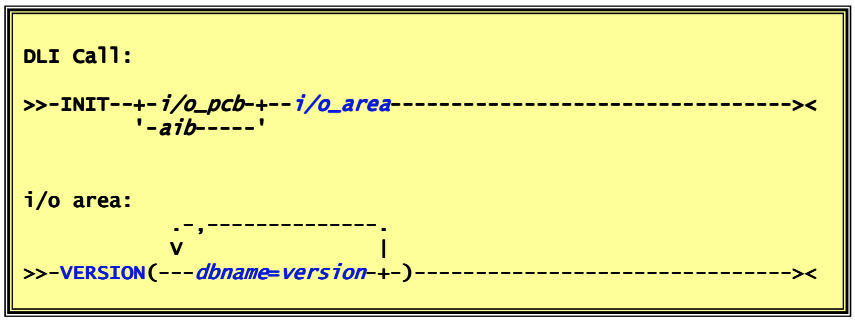
Calculate the size required for the I/O area by multiplying the number of databases that are specified in the input I/O area by 20.

For performance reasons, the INIT call should not be issued before the first GU call to the I/O PCB. If the INIT call is issued first, the GU call is not processed as efficiently.

Implementation – DL/I INIT Call

- I/O Area contains the 'VERSION' function and database parms
  - "VERSION(*dbnameA=version#,…,dbnameZ=version#,…)"
    - dbname: specifies a physical database name
    - version#: specifies a DBD version number to be used when accessing the database
    - sub-parameters separated with a comma
    - no duplicates allowed

INIT VERSION call and i/o area



i/o area - Specifies the I/O area in your program that contains the character string or strings indicating which INIT functions are requested. This parameter is an input parameter. INIT function character strings include DBQUERY, STATUS GROUPA, STATUS GROUPB, & VERSION (dbnameA=version#,...,dbnameZ=version#).

dbname: This specifies the database name.

version#: This specifies the version number of the database (identified in the DBname parameter) definition to be used when accessing the database.

The valid version range is 0 – 2147483647. When dbname is provided, version# is required.

1. Minimum requirement for a complete INIT VERSION call:
   - Open parenthesis '('
   - Database name. It must be alpha characters.
   - Equal sign '='
   - Database version number. It must be numeric.
   - Close parenthesis ')'
2. One set includes a database name, an equal sign, and a database version number.
3. Allows more than one set within () for INIT VERSION call.
4. Requires open and close parenthesis.
5. Requires equal sign as a separator between the database name and the database version number.
6. Requires a comma as a delimiter between the sets.
7. Allows no space between the database name, equal sign, database version number, and comma within the ().

   For example: INIT VERSION (DBname1=10,DBname2=20,DBname3=30)
8. Duplicate database name is not allowed within the call.

New INIT Call Status Codes

- "BE" Status Code
  - Explanation
    - A database name specified on the VERSION function of the INIT call can't be found
  - Programmer response
    - Correct the database names in the I/O area of the VERSION function
- "BG" Status Code
  - Explanation
    - The database type of the database named on the VERSION function of the INIT call does not support database versioning
    - The database types that do not support database versioning include:
      - GSAM
      - Logical databases
      - MSDB
  - Programmer response
    - Check that the database name is specified correctly

08- IMS 13 DB & DBRC: 520

New INIT Call Status Codes (cont'd)

- "BF" Status Code
  - Explanation
    - The INIT VERSION call is invalid
      - Invalid database version number specified in I/O area of INIT VERSION call
        - not a version number of the specified database
        - not within the supported range of values
        - different from the version number specified on a previous INIT VERSION call issued by the application program for the same database
      - Database versioning is not enabled in the IMS system
  - Programmer response
    - Check that the database name and version number are specified correctly in the INIT VERSION call
    - Confirm that database versioning is enabled in the IMS system
      - If not, remove the INIT VERSION call from the application program or coordinate with the SYSPROG or DBA to enable database versioning

08- IMS 13 DB & DBRC: 521

Changed INIT Call Status Codes

Status Code	Description
AG	During INIT VERSION call processing, the I/O area was not large enough to contain all of the output data; the output data was truncated to fit in the I/O area
AJ	For INIT VERSION (*dbname=number*) call, the syntax of the parameters in the I/O area is invalid
BA	Application programs that issue the INIT STATUS GROUPA call can receive a BA status code when database versioning is enabled and an invalid database version has been specified
NA	For an INIT VERSION(*dbname=version*) call, one or more of the databases that can be accessed by using this PCB is not available
SA	For an INIT VERSION(*dbname=version*) call, IMS could not get CSA storage to build the required internal blocks

08- IMS 13 DB & DBRC: 522

Database Version Determination

- If database versioning is enabled, the database version used to return IMS data to a program is determined as follows:
  - ✓ **DFSDFxxx DATABASE Section DBLEVEL=** parameter
  - ✓ **PSBGEN Statement DBLEVEL=** parameter
  - ✓ **PCB Statement DBVER=** parameter
  - ✓ **DL/I "INIT VERSION"** call

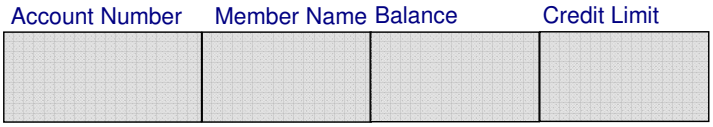
DL/I INIT VERSION call takes precedence over all other database version number specifications and defaults

Hierarchy for setting of database versioning.

DLI INIT VERSION call is always the overriding version specification .

Versioning Example

"CIF" Customer Account Segment

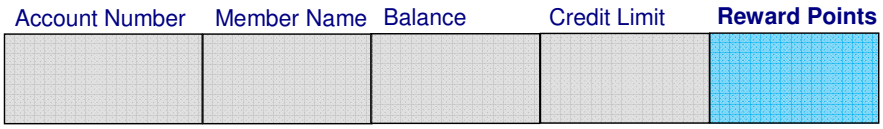


- Database Versioning enabled with DBLEVEL=BASE
- **BASE** version of the customer account segment
- Segment is fixed length
- No space remaining for additional fields

08- IMS 13 DB & DBRC: 524

Field Added to Existing DB Segment

"CIF" Customer Account Segment



- New version of the customer account segment defined in DBD
  - Expands the length of the segment
  - New field called Reward Points defined in expanded space
  - CIF DBD defined as DBVER=201401
- Existing applications are not updated
  - Existing applications do not have to know the new field exists
  - Existing applications do not see or update the new field

Segment READ/REPLACE Behavior

- Application scheduled using the **BASE** version of the database
  - Read a "CIF" Customer Account segment
  - Update the Balance to 200,000 with a REPL call
    - Only the credit balance field is changed
    - Reward Points field and others are left unchanged

Customer Account segment before replace

Account Number	Member Name	Balance	Credit Limit	Reward Points
555555	Vern Watts	100,000	500,000	50,000

Customer Account segment after replace

Account Number	Member Name	**Balance**	Credit Limit	Reward Points
555555	Vern Watts	**200,000**	500,000	50,000

08- IMS 13 DB & DBRC: 526

Segment READ/REPLACE Behavior

- Application scheduled using the **201401** version of the database
  - Read a "CIF" Customer Account segment
  - Update the Balance to 250,000 and Reward Points to 100,000 with a REPL call
    - Both the Balance and the Reward Points fields are changed
    - Others fields are left unchanged

Customer Account segment before replace

Account Number	Member Name	Balance	Credit Limit	Reward Points
555555	Vern Watts	200,000	500,000	50,000

Customer Account segment after replace

Account Number	Member Name	**Balance**	Credit Limit	**Reward Points**
555555	Vern Watts	**250,000**	500,000	**100,000**

08- IMS 13 DB & DBRC: 527

Database Versioning Operational Considerations

- When an application program requires a database version not currently the active version:
  - IMS will go to the Catalog to get the information for the requested DBD version
  - IMS will compare the Catalog info for the DBD to the currently active DBD information and determine the differences
  - If the IMS Catalog is not enabled, an “NA” status code is returned because the catalog database is not available
- When database versioning is enabled and the application program does not request a version:
  - By default IMS will retrieve the database data at the current physical level
  - Unless the DFSDFxxx default behavior is overridden with the DBLEVEL=BASE parameter

During scheduling, if a specific DB version is requested on the PCB and it is not the current version:

IMS will go to the catalog to get the correct DB version

If the IMS catalog is not enabled

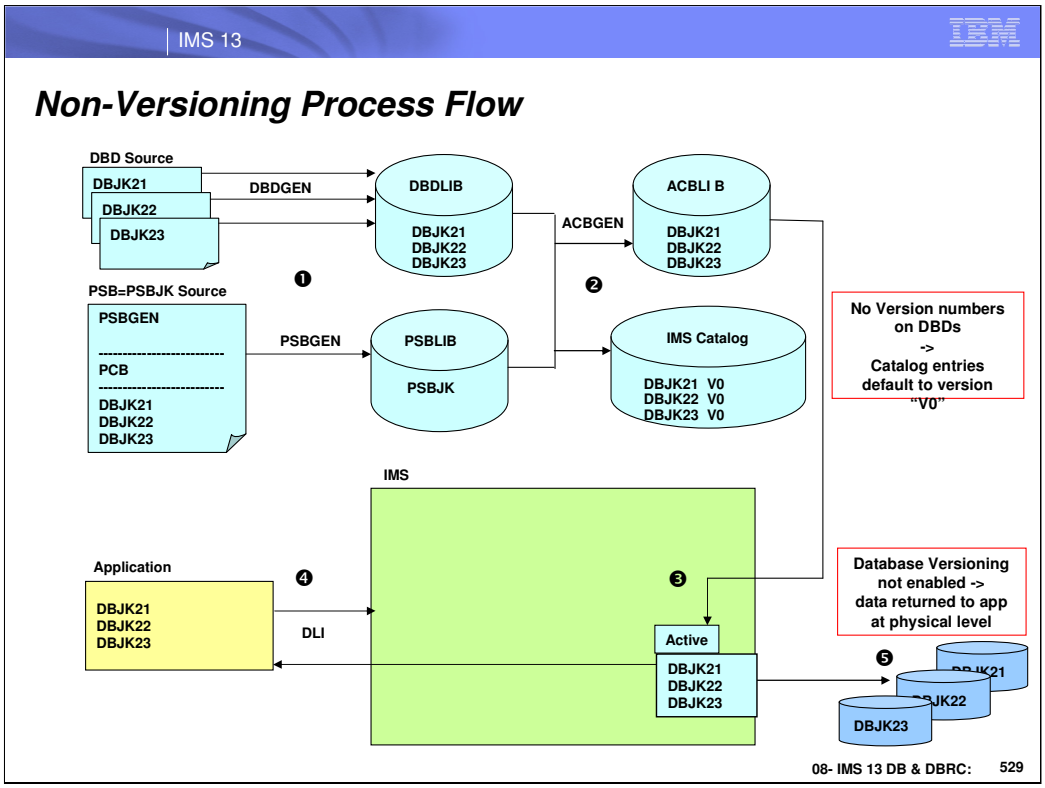
DLI call using catalog PCB DFSCAT00 fails because the catalog PSB DFSCP000 could not be scheduled

the DBPCB status code will be set to “NA”

If an application does not request a specific DB version:

IMS will retrieve the DB data at the current physical level

UNLESS the DBLEVEL= parameter has been set on the PSBGEN statement or in the DATABASE section of the DFSDFxxx PROCLIB member.



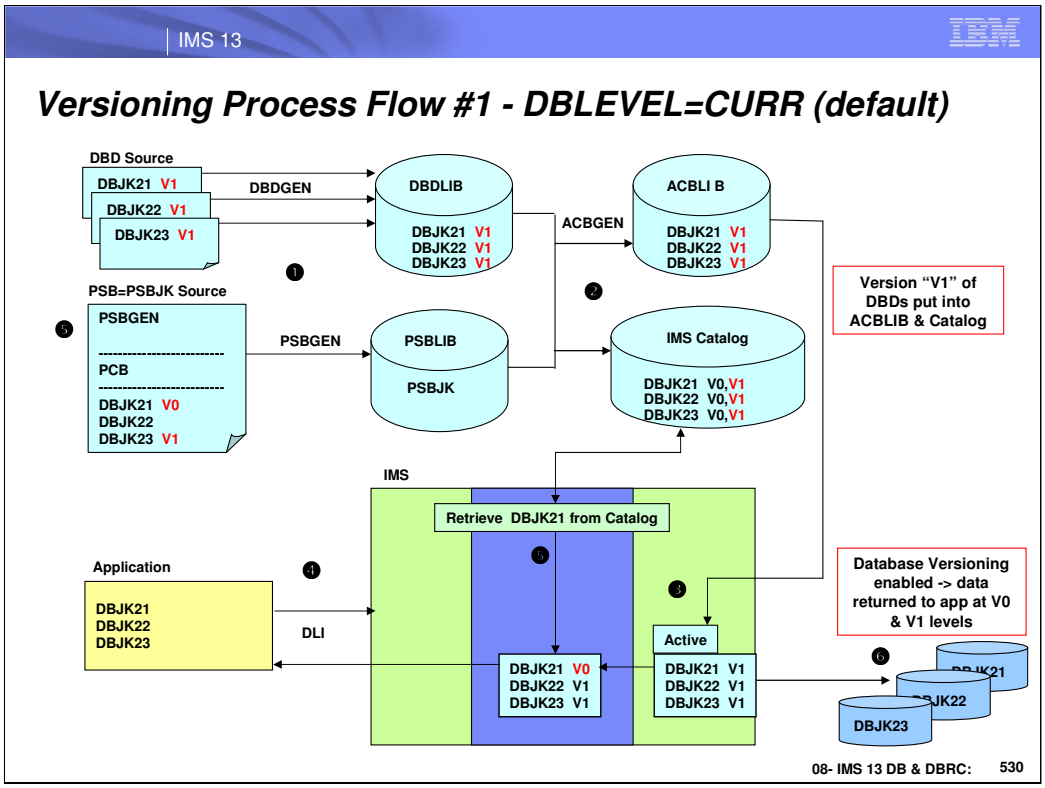
Current flow of how IMS Database works today without Database Versioning:

1. & 2. DBDGEN/PSBGEN/ACBGEN with no version numbers specified on the DBDs. If no version number specified, Catalog saves entries for the DBDs with no version number (version 0).

3. The active versions of DBJK21,DBJK22, & DBJK23 DBDs do not contain version numbers, therefore, Database Versioning is not enabled

4. Application does DB calls to access DBJK21, DBJK22 & DBJK23 databases.

5. IMS data is returned to application at the current physical structure level.

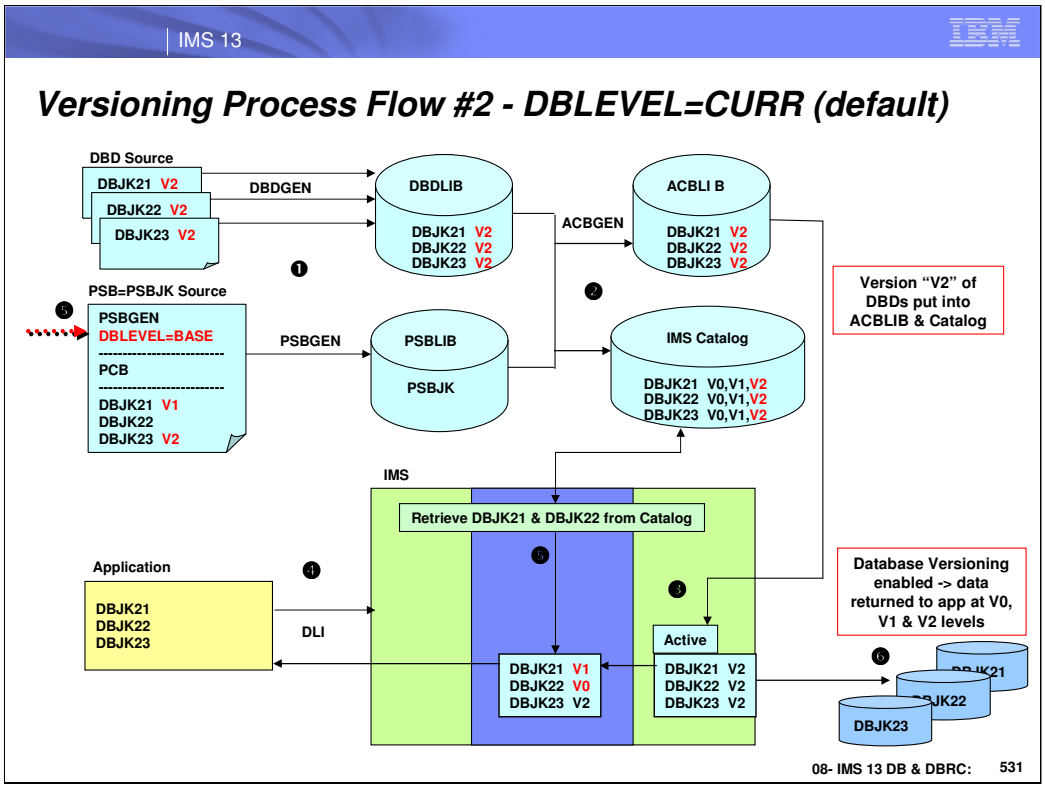


Once you enable Database Versioning and start specifying DBVER on the DBD source.

1. DBD NAME=DBJK21 DBVER=1…
   DBD NAME=DBJK22 DBVER=1…
   DBD NAME=DBJK23 DBVER=1…

   DBDGEN/PSBGEN/ACBGEN

2. Version 1 of DBJK21, DBJK22, & DBJK23 gets into ACBLIB.
   Version 1 of DBJK21, DBJK22, & DBJK23 gets into IMS Catalog in addition to the already existing Version 0.
3. IMS runs with Version 1 as the active version.
4. Application accesses DBJK21,DBJK22, & DBJK23.
5. The PSB indicates the application needs DBJK21 at Version 0 (Base version). IMS retrieves version 0 from the IMS Catalog.
   By default, the application uses DBJK22 at active Version 1.
   The PSB indicates the application will use DBJK23 at active Version 1.
6. DBJK21 IMS data is returned to application at V0 level.
   DBJK22 IMS data is returned to application at V1 level.
   DBJK23 IMS data is returned to application at V1 level.

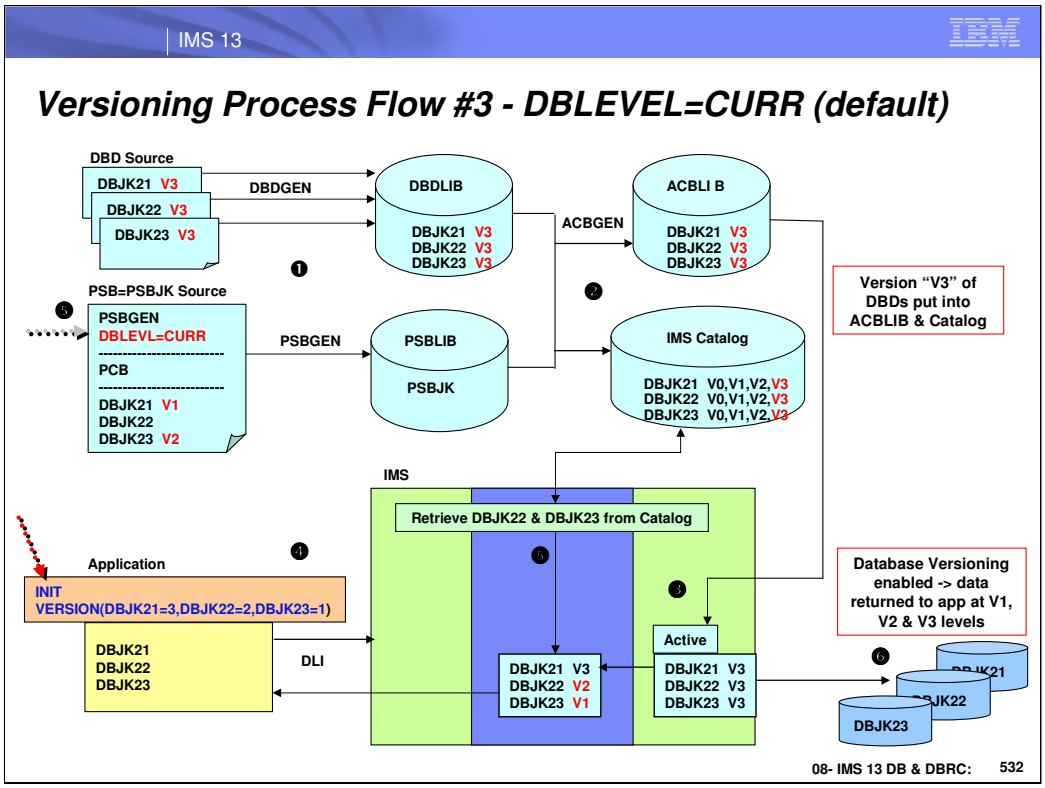


Once you enable Database Versioning and start specifying DBVER on the DBD source.

1. DBD NAME=DBJK21 DBVER=2…
   DBD NAME=DBJK22 DBVER=2…
   DBD NAME=DBJK23 DBVER=2…

   DBDGEN/PSBGEN/ACBGEN

2. Version 2 of DBJK21, DBJK22, & DBJK23 gets into ACBLIB.
   Version 2 of DBJK21, DBJK22, & DBJK23 gets into IMS Catalog in addition to already existing Version 0 & 1.
3. IMS runs with Version 2 as the active version.
4. Application accesses DBJK21,DBJK22, & DBJK23.
5. The PSB indicates the application needs DBJK21 at Version 1 & DBJK23 at Version 2 which is active in IMS.
   The PSB also indicates (DBLEVEL=BASE) meaning the default behavior is to use the base DBD version if one is not specified on the PCB.
   DBJK22 uses Version 0 from the IMS Catalog
6. DBJK21 IMS data is returned to application at V1 level.
   DBJK22 IMS data is returned to application at V0 / BASE level
   DBJK23 IMS data is returned to application at V2 level



Once you enable Database Versioning and start specifying DBVER on the DBD source.

1. DBD NAME=DBJK21 DBVER=3…
   DBD NAME=DBJK22 DBVER=3…
   DBD NAME=DBJK23 DBVER=3…

   DBDGEN/PSBGEN/ACBGEN
2. Version 3 of DBJK21, DBJK22, & DBJK23 gets into ACBLIB.

   Version 3 of DBJK21, DBJK22, & DBJK23 gets into IMS Catalog in addition to already existing Version 0,1&2
3. IMS runs with Version 3 as the active version.
4. Application accesses DBJK21,DBJK22, & DBJK23.
5. The application specified the INIT VERSION call which overrides all the DBD versions indicated on the PSB.

   DBJK21 uses Version 3 which is active in IMS.

   DBJK22 uses Version 2 from the IMS Catalog.

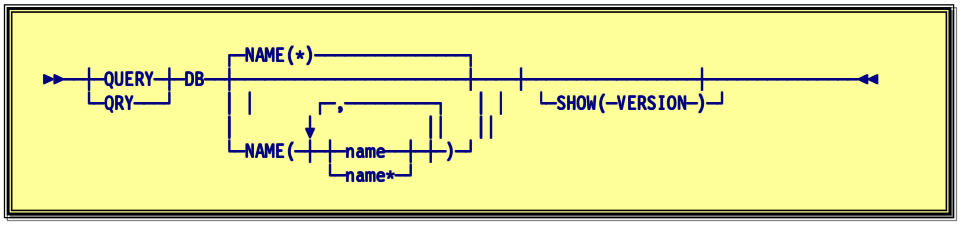
   DBJK23 uses Version 1 from the IMS Catalog.
6. DBJK21 IMS data is returned to application at V3 level.
   DBJK22 IMS data is returned to application at V2 level
   DBJK23 IMS data is returned to application at V1 level

.

Enhanced Type-2 QUERY Command

- QUERY DB
  - New SHOW(VERSION) option
  - Returns the version number of a database that is currently active in the online IMS system
    - For HALDB Master – response shows master and each partition
    - For HALDB Partition – response shows only the requested partition
    - For FP DEDB – response show only the DEDB
  - Cannot specify the VERSION filter with other SHOW filters

QUERY Command with SHOW(VERSION) option



VERSION

Returns the version number of the version of a database that is currently active in the online IMS system. The active version of a database is the version that is stored in the database control blocks that are loaded by the online IMS system. The control blocks define the actual physical structure of the database to the online IMS system.

The possible version numbers range from 0-2147483647.

You cannot specify this filter with other SHOW filters; you must specify SHOW(VERSION) individually.

SHOW(VERSION) is valid only for the following database access types:

- HDAM
- HIDAM
- PHDAM
- PHIDAM
- DEDB

If SHOW(VERSION) is specified for an unsupported database access type, completion code BD is returned in the CC column of the output to indicate that the query is invalid for the access type of the database.

If SHOW(VERSION) is specified for a HALDB master, the output lists a response line for the HALDB master name and for each of its partition. Each partition of the HALDB inherits the version number from the HALDB master, so each response line displays the same version number.

If SHOW(VERSION) is specified for a HALDB partition, the output lists a response line for just that partition. The version number that is displayed is that of the HALDB master. If the partition is disconnected from the HALDB master, such as might happen when the /DBR command is in effect on the database, completion code 10 is returned in the CC column of the output to indicate that no resource was found.

If SHOW(VERSION) is specified for a DEDB, the output lists a response line for only the DEDB. Area information is not listed in the output response.

Enhanced Type-2 QUERY Command

- QUERY DB Output

Short Label	Long Label	SHOW Parameter	Meaning
VER	VERSION	*VERSION*	Version number of the database that is currently active on the IMS system.
CC	CC	*Error*	Completion code. The completion code indicates whether or not IMS was able to process the command for the specified resource. Refer to the table of QUERY DB return, reason, and completion codes for more information. The completion code is always returned.
CCTXT	CCText	*Error*	Completion code text that briefly explains the meaning of the non-zero completion code. This field is returned only for an error completion code.

- **Short Label:** short label generated in the XML output
- **Long Label:** column heading for output field in the formatted output
- **SHOW Parameter:** parameter on the QUERY SHOW() keyword that caused the output field to be generated
  - *Error* appears for output fields that are returned for a non-zero completion code
- **Meaning:** a brief description of the output field

08- IMS 13 DB & DBRC: 534

Enhanced Type-2 QUERY Command

- QUERY DB Output
  - Output headers for successful commands

```
QUERY DB NAME(dbname) SHOW(VERSION)
DBName          PartName     MbrName     CC    TYPE       VERSION
--------        --------     -------     --    -----      ----------
```

  - Output headers for commands with errors

```
QUERY DB NAME(dbname) SHOW(VERSION)
DBName          PartName     MbrName     CC    CCText               TYPE       VERSION
--------        --------     -------     --    ------               -----      ----------
```

Completion Code	Completion Code Text	Meaning
0		Command completed successfully for the resource.
10	Resources not found	• The resource name is unknown to the client that is processing the request. The resource name might have been typed in error or the resource might not be active at this time. Confirm the correct spelling of the resource name specified on the command. • For full function databases that have not been opened or initialized. • If the command is used to query the version number of a HALDB partition, the output will list a response line for just that partition. If the partition is disconnected from the HALDB master, possibly due to a DBR of the database, then the output will show a completion code of '10' in the CC column to indicate no resource found.
195	Unsupported DB type	Database versioning for this database access type is not supported.

08- IMS 13 DB & DBRC: 535

Type-2 Query Command Security

- RACF Security definitions

IMS Command	Command Keyword	RACF access authority	Resource name
QUERY	DB	QUERY	IMS.plxname.QRY.DB

Use standard RACF definitions to secure the IMS command QUERY DB

Database Versioning Errors

- U3303 PseudoAbend
  - A DL/I call to access a specific version of the database data failed
    - Database Versioning enabled
      - invalid database version number was specified
      - requested database version cannot be found in the IMS catalog
      - current database structure contains a change that is not supported by versioning
      - storage error occurred while building the internal blocks required to satisfy a request for a version of a database other than the current version
    - Database Versioning not enabled
  - DFS3303I message issued to console before the U3303 abend
  - If INIT STATUS GROUPA was issued prior to DL/I call
    - U3303 pseudoabend is suppressed and a “BA” status code is returned

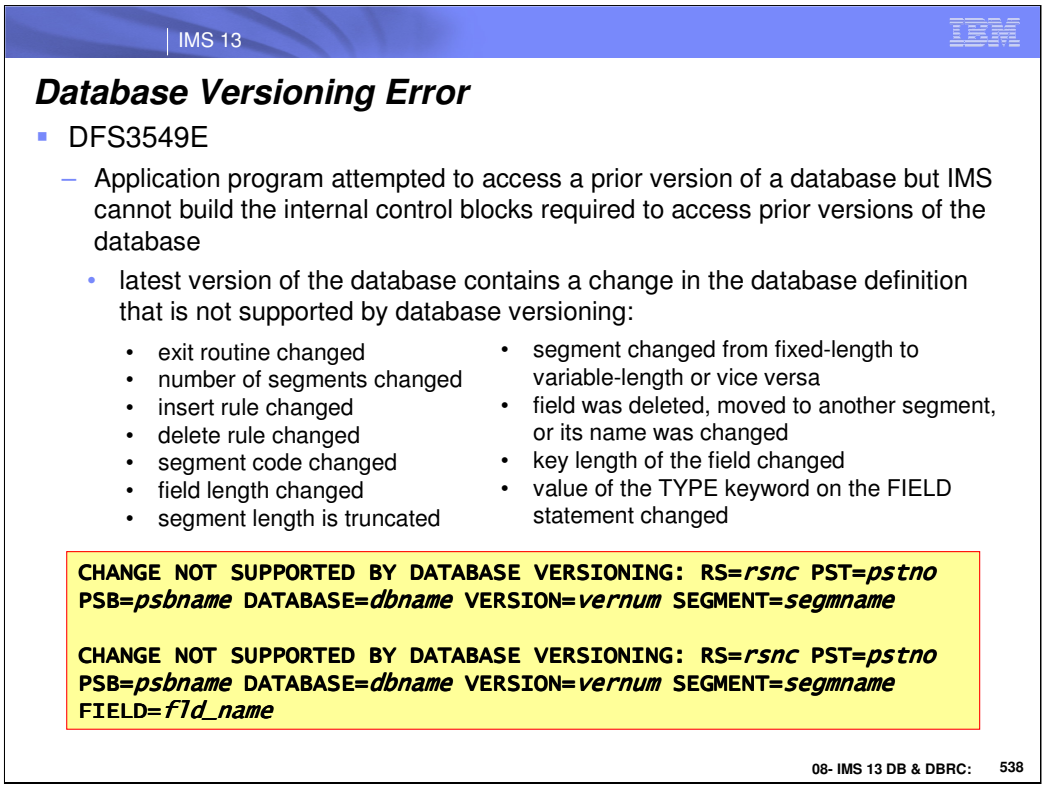
DFS3303I

PSB *psbname* PCB *pcbname* DBD *dbdnamexxxx* JOBNAME *jobname*RGN *nnn*

Explanation

This message precedes pseudoabend 3303 when an application program scheduled with PSB *psbname* tries to make an incompatible DL/I call to database PCB *psbname*. During DL/I scheduling of the PSB, database *dbdname* had condition *xxxx*. Depending on the condition, DL/I calls to this database are partially or totally restricted.

This message precedes only those 3303 abends caused by DL/I attempts to access data in a database that was unavailable when the program was scheduled, and the program had not issued the DL/I INIT call.



Explanation

An application program attempted to access a prior version of a High Availability Large Database (HALDB), but IMS cannot build the internal blocks that are required to access prior versions of the database, because the latest version of the database contains a change in the database definition (DBD) that is not supported by database versioning.

Application programs cannot access any prior version of the database, unless the application programs are changed or the unsupported change is removed from the database.

System action

IMS cannot build the internal blocks that are required to provide access to prior versions of the database. Only the most recent version of the database can be accessed.

IMS returns a status code to the application program or the application code abends.

System programmer response

Determine whether you need to keep the changes in the database or remove them.

Keeping the changes requires all application programs to be modified to access the new database structure. The prior versions of the database cannot be accessed anymore.

Removing the changes allows you to enable database versioning. However, if the database has been updated since the changes were made, you need to recover the physical database to the prior version. Any updates to the database that were made by application programs that used the current version of the database are lost.

Database Versioning Error Messages

- Enhanced IMS Messages

Message	Description
DFS3303I	New conditions: **INVDBVER** An invalid number for a full-function database was specified on a PCB or an INIT VERSION call that was issued by the application program. The specified database version number must be equal to or less than version number of the current database that is active in the IMS system. Also, a database version cannot be specified on a PCB if database versioning is not enabled. Database versioning is enabled by specifying DBVERSION=Y in the database section of the DFSDFxxx PROCLIB member. **INVDBCHG** The current database structure of a full-function database contains a change that is not supported by database versioning. Prior versions of the database are incompatible with the current version and can not be accessed. The changes that are supported by database versioning are: • Increasing the size of a segment • Adding new fields without changes made to existing fields **NOCATALG** The IMS catalog is not enabled. Database versioning requires the IMS catalog. **NOSTORAG** A storage error occurred while building the internal blocks that are required to satisfy a request for a version of a full-function database other than the current version. **NOVERFND** The requested version of a full-function database cannot be found in the IMS catalog
DFS3549E	Database change not supported by versioning: An application program attempted to access a prior version of a High Availability Large Database (HALDB), but IMS cannot build the internal blocks that are required to access prior versions of the database, because the latest version of the database contains a change in the database definition (DBD) that is not supported by database versioning. Application programs cannot access any prior version of the database, unless the application programs are changed or the unsupported change is removed from the database.

08- IMS 13 DB & DBRC: 539

Database Versioning Error Messages

- New IMS Messages

Message	Description
DFS0006E	An error was detected while attempting to load the data management block (DMB) of an altered database.
DFS0123E	An application program attempted to access a prior version of a Fast Path data entry database (DEDB), but IMS cannot build the internal blocks that are required to access prior versions of the database, because the latest version of the database contains a change in the database definition (DBD) that is not supported by database versioning
DBD180	The DBVER operand is specified on a DBD statement for a database access type that does not support database versioning.
DBD181	The value on the DBVER operand in the DBD statement is not valid.
PCB540	The value on the DBVER operand of the PCB statement was not valid.
PGEN259	The value on the DBLEVEL operand of the PSBGEN statement was not valid

08- IMS 13 DB & DBRC: 540

Database Versioning Migration Considerations

- Populate the IMS catalog with ACBLIB definitions
- Enable the IMS catalog
  - Before Database Versioning is enabled for a database, IMS continues to only recognize the current physical database definition
- All IMS systems in an IMSplex must be running IMS 13
- Enable IMS Database Versioning
  - Specify the new parameter to enable a default database versioning
    - Use the default DBLEVEL=CURR setting in DFSDFxxx so all applications access databases at the latest, physical DB version
    - Set the DBLEVEL=BASE parameter in DFSDFxxx so all applications access databases at the oldest, lowest DB version
  - Use the PSBGEN DBLEVEL parm or the PSB PCB DBVER= parm if an application needs to use a version of a database different from the system default level
  - Use DLI INIT VERSION call is needed for dynamic version switching

08- IMS 13 DB & DBRC: 541

Database Versioning Summary

- IMS 13 allows application programs to use different versions of the same physical database
  - Multiple views of the physical data are maintained in the IMS catalog
  - Application programs can use different views of the same physical IMS database
- Benefit
  - Customers can support multiple versions of an IMS database
  - Physical database structure can be changed without having to modify all the existing application programs using the database

Database Versioning Support is for Full Function, HALDB and DEDB database customers who need support for multiple views of the physical data to a variety of application needs such as:

•Implementing application changes over time.

•Ability to use application programs, for which there is no source code, after database structure changes.

Versioning support enables users to assign user-defined version IDs to different versions of the structure of a database. The user-defined version IDs are stored in the record for the database in the IMS Catalog. Upon accessing the database, application programs specify the version of the database that they need. If they do not specify a version, by default they will get the version of the database structure at the current level.

HALDB Alter

08- IMS 13 DB & DBRC: 543

HALDB Alter

- IMS 13 provides ability to make structural changes to a HALDB database without a database outage
  - Structural changes can be made to DB segment definitions
  - Online Reorganization is used to apply the structural changes to the online HALDB database
  - Online Change process is used to activate the new ACBLIB member(s) in the online IMS system
- Benefit
  - Eliminate a database outage when structural changes must be made to segment definitions in a DBD
  - Improved online availability of HALDB databases

HALDB Alter is for HALDB database customers who want to have the ability to make segment changes without unloading and reloading the database.

This addresses the challenge of maintaining database availability while changing the structure of a HALDB database. The actual hierarchy of the IMS HALDB cannot be changed.

When a segment change is made to a DBD, an online command can be issued to apply the change to the database. The change is implemented via an option of the HALDB Online Reorganization (OLR) function. Application programs can access the database at the same time the OLR function is changing the structure of the database.

This line item provides value to customers by reducing the complexity of making structural changes to a HALDB database, eliminating system down time, and improving system availability. This line item also reduces the cost and risk associated with making and coordinating wholesale changes to all application programs when database structure changes occur. This line item allows customers to improve their HALDB database structures even if the database is used by critical application programs that they no longer have the ability to change.

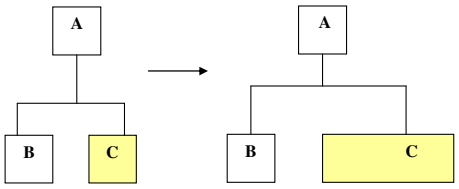
HALDB Alter Prerequisites

- Software requirements
  - Same as IMS Version 13
  - CSL
    - SCI
    - OM
- Hardware requirements
  - Same as IMS Version 13

Minimal software and hardware pre-reqs:
IMS 13 and CSL (SCI and OM)

HALDB Alter Overview

- Structural changes can be made to online HALDB database segments
  - PHDAM
  - PHIDAM
- Types of structural segment changes
  - Add a new field to space at the end of an existing segment
  - Define new fields to remap fields and space in an existing segment
  - Increase the length of an existing segment



The following structural changes can be applied to an online HALDB with the type-2 INIT OLREORG command:

Increasing the length of existing segment.

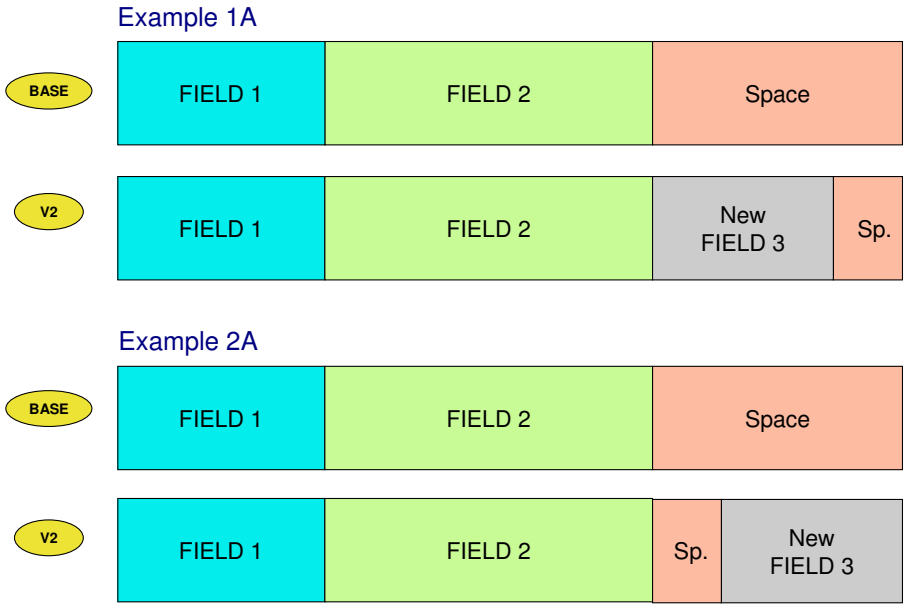
Adding new fields to space at the end of the segment.

Defining new fields that redefine existing fields and space in the segment.(not structural)

When a segment change is made to a DBD, an online command can be issued to initiate the change to the database. The change is implemented via an option of the TYPE 2 HALDB Online Reorganization (OLR) function.

Application programs that use the old database definitions can access the database while the OLR function is altering the structure of the database.

Add a new field to space at the end of a segment

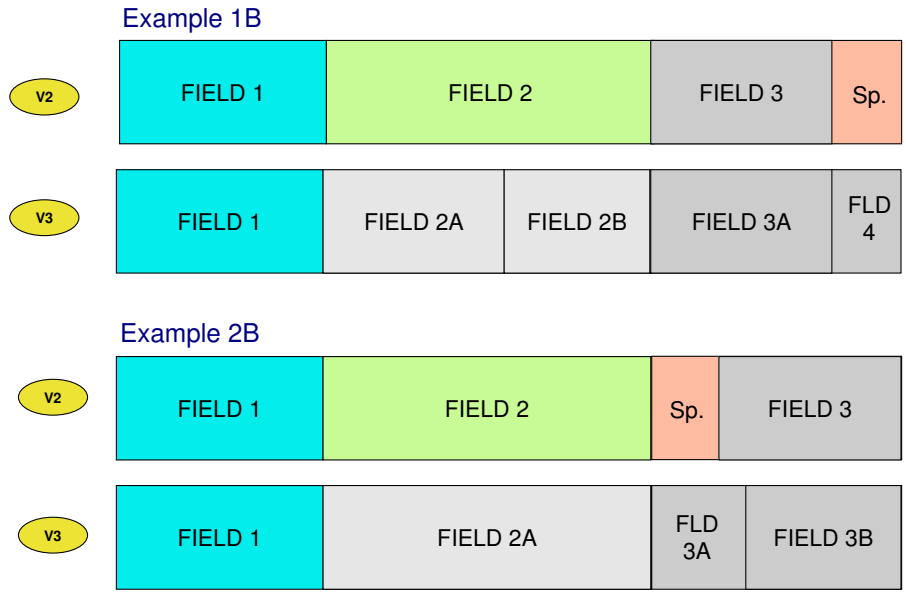


EX 1A:
Add new field to beginning of free space at end of segment

EX 2A:
Add new field to end of free space at end of segment

In addition to the existing field definitions, the new fields are added to the DBD segment definition.

Define new fields to remap fields & space in a segment



548

EX 1B:

Define 2 new fields, FIELD 2A and FIELD 2B, to overlay/re-map FIELD 2, define new FIELD 3A to remap FIELD 3, and define FIELD 4 in free space.

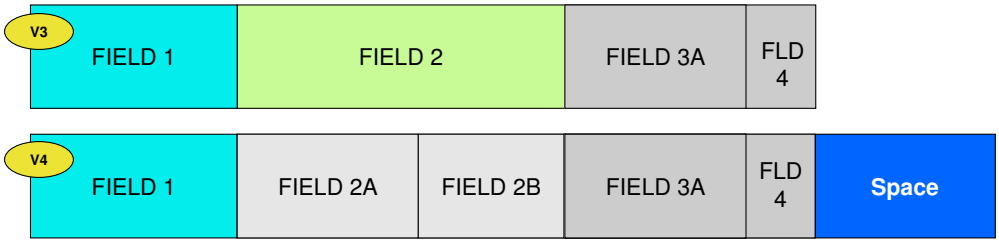
EX 2B:

Define new field, FIELD 2A, to overlay/re-map FIELD 2, and define new FIELD 3A and FIELD 3B to remap free space and FIELD 3.

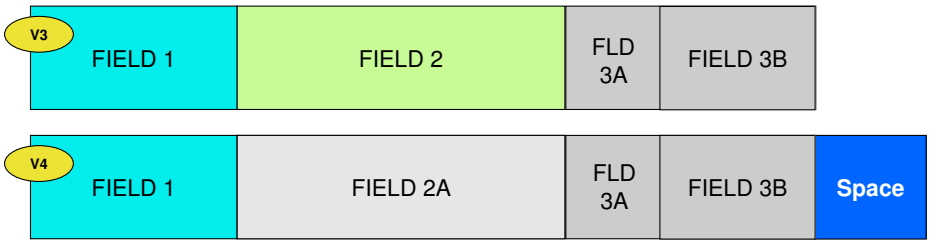
In addition to the existing field definitions for the segment. The new overlay fields are added to the DBD segment definition.

Increase the length of an existing segment

Example 1C



Example 2C



EX 1C:
Increase the length of an existing segment by adding space at end

EX 2C:
Increase the length of an existing segment by adding free space at end

The new free space is added to the segment size in the DBD definition.

HALDB Alter Overview

- After changes are made to a DBD segment definition(s)
  - Online Reorganization (OLR) is used to ALTER the online database from the current structure to the new structure
    - DB unload / reload not required
    - Requires Type-2 INITIATE OLREORG
      - /INITIATE OLREORG does not support altering a HALDB
  - Online Change (OLC) is used to activate the changed ACBLIB members in the IMS online system
  - Application programs can start using the new database structure

When a segment change is made to a DBD, an online command can be issued to initiate the change to the database. The change is implemented via an option of the TYPE 2 HALDB Online Reorganization (OLR) function.

An Online Change must be completed to in-affect the DBD change in the online system.

Application programs that use the old database definitions can access the database while the OLR function is altering the structure of the database. Once the OLC is completed, application programs can use the new database definitions to access the database.

The TYPE-1 /INIT OLR command will not support the new HALDB ALTER function.

HALDB Alter Preparation

- Modify the DBD source code
  - Define new fields in space at the end of segment(s)
    - Specify new FIELD statements
  - Define new fields to remap existing fields & space in a segment(s)
    - Specify new FIELD statements
  - Increase the length of a segment(s)
    - Specify new length in the BYTES= parameter of SEGM statement
- Run DBDGEN
  - Modified DBD source used as input
- Run PSBGEN(s) (if needed)
  - Modified PSB source used as input
- Run ACBGEN
  - Create a new member in a **staging** ACBLIB
- Modify or code new application programs to use new fields

Begin the ALTER process by making coding changes to the DBD source. Multiple segment definitions in one DBD can be changed at the same time.

The appropriate DBD and ACB GENs must be run. The ACB member should be genn'd to an output staging ACBLIB.

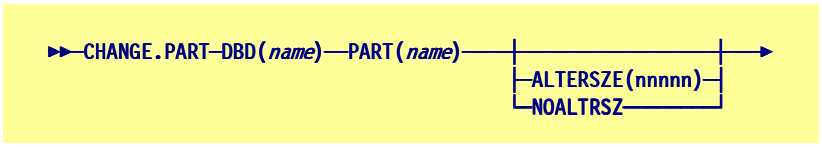
The staging ACB library needs a dynamic allocation member – DFSMDA.

A DFSMDA member can be created for the staging ACBLIB if one doesn't already exist. Use the documented DFSMDA macros.

During ALTER processing, IMS will process all the changes made in the DBD that are found in the staging ACBLIB for that DB – all segment changes are made at one time.

HALDB Alter Preparation - DBRC

- CHANGE.PART Command
  - Use to set attributes for a HALDB partition before it is altered
    - New ALTERSZE keyword to set block/CI size of o/p partition data sets
    - New NOALTRSZ keyword to clear block/CI size of o/p partition data sets
    - Sizes cannot be changed if the HALDB is currently being altered
      - Only after the alter operation is completed and an online change is performed



If you are increasing the size of a segment when you are altering an online HALDB database, you might also need to increase the OSAM block size or VSAM CI size of the output database data set that holds the altered segment.

New block or CI sizes are applied to the output data sets at the start of alter processing, but must be entered in the RECON data set before the INIT OLREORG OPTION(ALTER) command is issued.

New block or CI sizes are entered into the RECON by specifying them on the ALTERSZE keyword of the CHANGE.PART command or by specifying them in the Change Dataset Groups panel of the HALDB Partition Definition utility (%DFSHALDB).

For VSAM data sets, if output data sets for alter processing exist, the output data sets that require a new CI size must be deleted before initiating the alter process. Alter processing automatically re-creates the required output data sets with the new CI size. If no ALTERSZE value is specified for a given VSAM data set group and an output data set exists, the CI size of the output data set is used. If no ALTERSZE value is specified and an output data set does not exist, the CI size of the input data set is used.

For OSAM data sets, if no ALTERSZE value is entered, the BLKSZE of the input data set is used, even if an output data set exists.

When you change a block or CI size, you might also need to change the size of the buffers. If the new block or CI size does not fit into the current buffer subpool, IMS tries to find a larger subpool among the available subpools. If none of the available subpools are large enough to hold the new block or CI size, the output data set fails to open. To check buffer sizes, issue the type-2 command QUERY POOL TYPE(DBAS).

HALDB Alter Preparation - DBRC

- CHANGE.PART Command
  - **ALTERSZE(*nnnnn*) | NOALTRSZ**
    - Mutually exclusive, optional keywords
    - **ALTERSZE(*nnnnn*)**
      - Specifies new OSAM block sizes or VSAM CI sizes for the output partition data sets
      - Specify up to 10 values, one for each data set group defined in the DBD
        - Omitted values remain unchanged
        - Numeric values must be even and no greater than 32K
      - New sizes must be stored in the RECON partition record before the INITIATE OLREORG OPTION(ALTER) command is issued
      - New sizes are stored in the RECON partition record until the alter operation is complete and an online change is performed
      - After online change is performed, new OSAM block sizes are saved in the OSAM BLOCK SIZE field of the RECON partition record

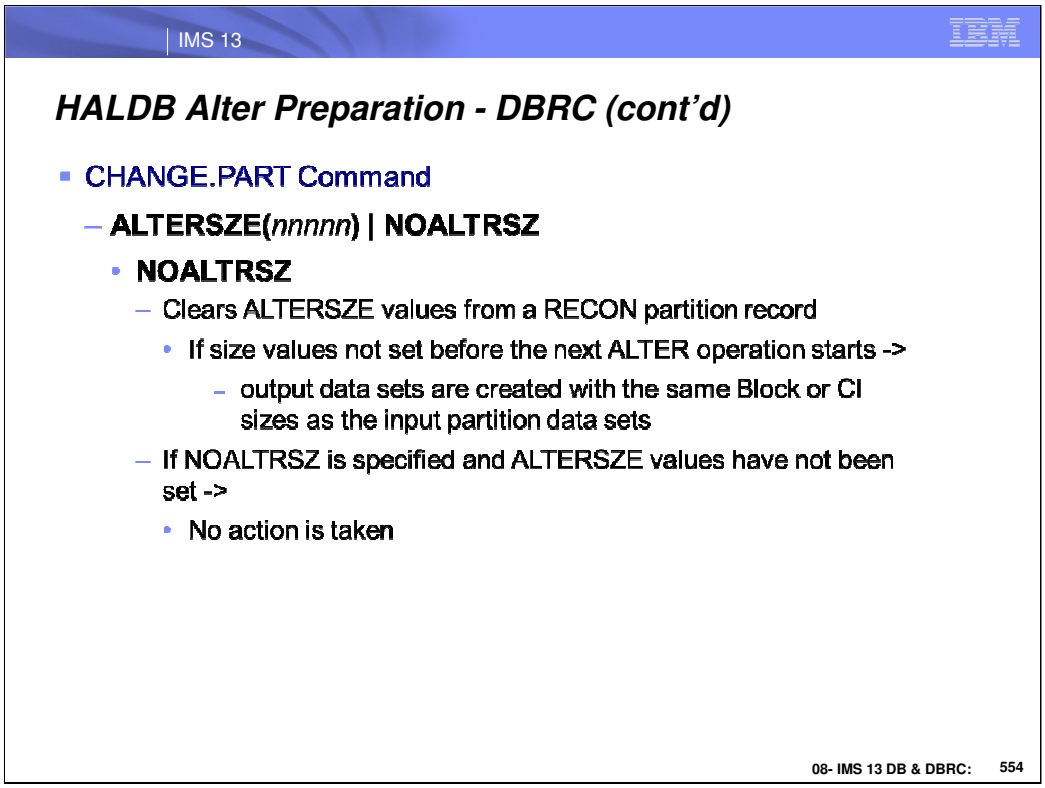
To increase the OSAM block size or VSAM CI size of the database data sets when you are modifying the structure of a database with the HALDB alter function, you must set ALTERSZE values for each data set group that is changing in each partition record in the RECON.

To set the ALTERSZE values, you can use either DBRC command CHANGE.PART or the HALDB Partition Definition utility (%DFSHALDB).

If you use the CHANGE.PART command to set the ALTERSZE values, the values must be specified as positional, comma-separated values. The value in the first position applies to the first data set group. The value in the second position applies to the second data set group, and so on.

For example, the following ALTERSZE keyword sets a new block or CI size for the third data set group, but leaves the sizes unchanged for the first and second data set groups, as well as for the fourth through tenth data set groups, if they exist: ALTERSZE(,,4096).

You can determine the position in which to enter a size for a data set group by looking at the DSGROUP keyword in the SEGM statement that defines the segment that you are altering. DSGROUP=A indicates the first position, DSGROUP=B, the second position, and so on up to DSGROUP=J, which indicates the tenth position.



To correct an ALTERSZE value that is already set, replace the incorrect value with the correct value by using either the CHANGE.PART command or the HALDB Partition Definition utility (%DFSHALDB).

For OSAM data sets, if you change an ALTERSZE value back to the original block size of the input data set, the ALTERSZE value displays as 0 to indicate that the block size is not changing. If all of the ALTERSZE values are restored to the original block sizes of the input data sets, the ALTER SIZE field is omitted from the output of the LIST.DB command.

For VSAM data sets, if you change an ALTERSZE value back to the original CI size, the original CI size is displayed. If all of the ALTERSZE values are restored to the original CI sizes of the input data sets, the ALTER SIZE field is displayed with the last values that you entered.

You can clear all of the ALTERSZE values for a partition by specifying CHANGE.PART PART(name) NOALTRSZ command. For both OSAM and VSAM data sets, when the NOALTRSZ keyword is used to clear all ALTERSZE values, the ALTER SIZE field is omitted when the partition record is displayed.

After the command is successfully processed, the block or CI sizes to be used by the alter process are listed under ALTER BLOCK SIZE in the RECON record for a partition, which can be displayed by the issuing DBRC command LIST.DB DBD(*partitionname*).

After the alter size values are corrected, you can start the alter process by issuing the IMS type-2 command INIT OLREORG NAME(*masterdb*) OPTION(ALTER).

HALDB Alter Online Process

- Use Online Reorg to ALTER the structure of a HALDB database
  - Applies structural changes to the online HALDB database
  - All database partitions are included in the reorg ALTER process
  - Reads the current DBD version of the active ACBLIB
  - Reads the new DBD from a staging ACBLIB
  - If necessary, 10 TCBs will be scheduled concurrently for ALTER
- Application programs accessing the existing database, using the current version of the DBD, continue running
- Internal tables are built to represent the changes between the active/input DMB and the staged/output DMB

When the ALTER option is specified, the INIT OLREORG command initiates a reorganization of an entire HALDB database to apply the database changes to all of the database partitions. During ALTER processing, IMS will process all the changes made in the DBD that are found in the staging ACBLIB for that DB – all segment changes are made at one time.

Upon receiving the INIT OLREORG OPTION(ALTER) command, an IMS system can alter up to 10 partitions concurrently. Any partitions that cannot be processed immediately are queued internally until they can be altered.

While an IMS system reorganizes and alters a partition, the IMS system has ownership of the partition. The subsystem ID of the IMS system that owns a partition for alter processing is recorded in the OLRIMSID field of the partition record.

In a data-sharing environment, ownership of a partition is granted to the first IMS system that is available to alter the partition. If one IMS system is available to process ten partitions before any other IMS system becomes available, all ten partitions are processed by the single IMS system. If partitions are queued for alter processing, ownership of the queued partition is granted to the first IMS system to be altering less than ten partitions concurrently.

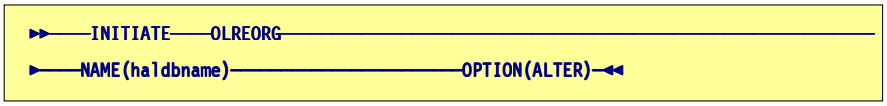
As soon as the INIT OLREORG OPTION(ALTER) command is received by IMS, DBRC marks every partition in the database with ALTER IN PROGRESS=YES, even those partitions that are queued. While an IMS system is actively altering a partition, the partition record shows OLREORG CURSOR ACTIVE=YES.

After alter processing is complete for a partition, the partition record shows ALTER COMPLETE=YES.

Only after all partitions in the database have a status of ALTER COMPLETE=YES can you perform online change to activate the new database structure. The online change function resets both the ALTER IN PROGRESS field and the ALTER COMPLETE field to NO.

HALDB OLR Alter Command

- INITIATE OLREORG



Keyword	Function
NAME()	• specifies the name of a HALDB master • only one HALDB master database name can be specified • you cannot use the wildcard character (*)
OPTION()	• Specifies the different options that will affect how HALDB online reorganization performs (all options are valid)
ALTER	• specifies that the command will initiate the database structural alter processing

The INITIATE OLREORG command is used to initiate the dynamic structural change processing for HALDB partitions. The structural changes of the HALDB partitions are made in the DBD statement and can be accompanied by the DBVER parameter to identify the version of the database structure change. When the INITIATE OLREORG command is issued, it reads the new definitions from the staging library and is used to construct the OLR output data set as if the entire database is being reorganized (ie., all the partitions of a HALDB).

The following keyword parameters are changed/added to the type-2 INITIATE OLREORG command.

NAME()

This keyword specifies the name of a HALDB master. Unlike the existing INITIATE OLREORG command (which PHDAM or PHIDAM HALDB partition names can be specified), only one HALDB master name can be specified here. You cannot use the wildcard character (*).

OPTION()

This keyword allows you to specify the different options that will affect how HALDB online reorganization performs.

ALTER

This option specifies that the command will initiate the database structural change processing. The new database definition is obtained from the staging library, so the new DBD for the database specified on the NAME() parameter needs to be genned into the staging library prior to issuing the command.

HALDB Alter Online Process

- Issue Type-2 Initiate Online Reorganization command

INITIATE OLREORG NAME(masterdb) OPTION(ALTER)

  - Initiates the dynamic structural change processing for HALDB partitions
  - Only one HALDB master database name can be specified per INIT command
  - IMS reads the new DBD definition from a staging ACBLIB
    - Staging ACBLIB is dynamically allocated
    - Member info is used to construct the OLR output data set
    - Output data sets for all the HALDB partitions are built
  - Type-2 TERM OLREORG or Type-1 /TERM command is allowed while database structure change is in progress
  - Type-2 INIT OLR with OPTION(ALTER) command will restart the HALDB ALTER structure process where it left off

You can stop alter processing of a HALDB database before it is complete by issuing the TERMINATE OLREORG command or /TERM command. You can issue either the type-1 or type-2 version of the TERMINATE OLREORG command; however, only the type-2 version of the command can be issued to multiple IMS™ systems. The type-1 command can only be processed on the IMS that 'owns' the OLR (ie., wherever the partition(s) are being re-organized).

The TERMINATE OLREORG command does not support the specification of the name of a HALDB master database. To stop alter processing for the entire HALDB database, you specify a wildcard character in place of the partition names or you can specify the names of all of the database partitions explicitly. If multiple IMS systems are altering the database, you **must** use the type-2 TERMINATE OLREORG command to stop all of the IMS systems at once or issue the type-1 command separately on each IMS system.

To stop alter processing for one or more partitions of a HALDB database, issue the command:

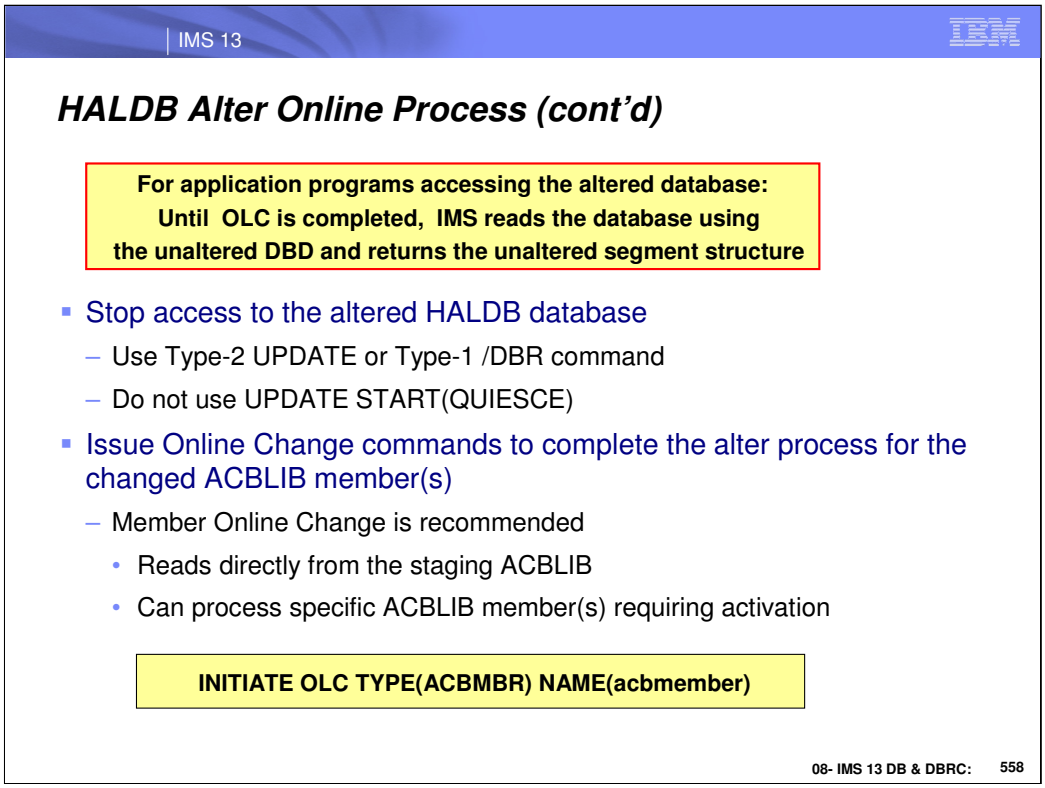
TERMINATE OLREORG NAME(*partnm* | *)

When alter processing is stopped for a subset of the partitions in the database, alter processing continues for the other partitions that are not contained within the specified subset.

When alter processing is stopped, the data in the partition might be physically stored in both the input and output data sets. The output data sets conform to the altered database structure. The input data sets conform to the old database structure. However, where the data is physically stored is not apparent to application programs. Until alter processing completes and online change is performed, application programs can access the data only in the old database structure.

To resume alter processing, issue the INITIATE OLREORG OPTION(ALTER) command.

To resume alter processing of a partition on a different IMS system, you can release the ownership of an IMS system by specifying the REL option. For example, INITIATE OLREORG OPTION(ALTER,REL)



The IMS™ online change function is required to enable access to the new structure of a HALDB database after alter processing completes.

Before you start the online change procedure to complete an alter operation, you must stop access to the HALDB database by issuing either:

/DBR DB *HALDB_master_name* command

UPDATE DB NAME(*HALDB_master_name*) STOP(ACCESS) command.

Do not use UPDATE START(QUIESCE)

In addition to activating the ACB members that contain the new database structure, the online change function clears various flags and counters in the RECON data set. Activating the ACB members by a means other than the online change function does not clear the flags and counters automatically.

Until the ACB members are activated and the flags and counters are cleared, the alter procedure is not complete and the new database structure cannot be used.

Use the member online change function to complete the alter procedure. The member online change function reads directly from the staging ACB library and can process only the specific ACB members that require activation. The local and global online change functions require you to copy the ACB members into the inactive ACB library. They also process the entire ACB library, instead of just the ACB members that contain the new database changes.

Until all of the partitions in the HALDB database are altered and online change is performed, only application programs that use the unaltered database structure can access the database.

HALDB Alter Online Process (cont'd)

- Start access to the altered HALDB database

New HALDB database structure can now be used by modified or new application programs needing the new segment fields

- Implement new or modified IMS application programs

The IMS online change function is required to enable access to the new structure of a HALDB database after alter processing completes.

Before you start the online change procedure to complete an alter operation, you must stop access to the HALDB database by issuing either:

/DBR DB *HALDB_master_name* command

UPDATE DB NAME(*HALDB_master_name*) STOP(ACCESS) command.

Do not use UPDATE START(QUIESCE)

In addition to activating the ACB members that contain the new database structure, the online change function clears various flags and counters in the RECON data set. Activating the ACB members by a means other than the online change function does not clear the flags and counters automatically.

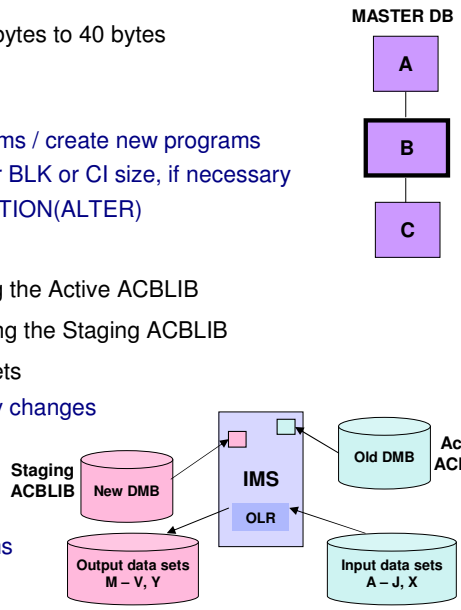
Until the ACB members are activated and the flags and counters are cleared, the alter procedure is not complete and the new database structure cannot be used.

Use the member online change function to complete the alter procedure. The member online change function reads directly from the staging ACB library and can process only the specific ACB members that require activation. The local and global online change functions require you to copy the ACB members into the inactive ACB library. They also process the entire ACB library, instead of just the ACB members that contain the new database changes.

Until all of the partitions in the HALDB database are altered and online change is performed, only application programs that use the unaltered database structure can access the database.

HALDB Alter Example

- Update DBD source for MASTER database
  - Increases the size of segment B from 30 bytes to 40 bytes
- Run DBDGEN
- Run ACBGEN(s) into a staging ACBLIB
- Make updates to affected application programs / create new programs
- DBRC CHANGE.PART ALTERSZE() to alter BLK or CI size, if necessary
- Issue INIT OLREORG NAME(MASTER) OPTION(ALTER)
  - IMS allocates Staging ACBLIB
  - IMS builds input DMB control blocks using the Active ACBLIB
  - IMS builds output DMB control blocks using the Staging ACBLIB
  - IMS allocates the output database data sets
- Backup Active ACBLIB members affected by changes
- Stop MASTER DB access
- Perform Online Change
- Start MASTER DB access
- Implement new/updated application programs



Walkthru example showing how to make a structural change to the MASTER DB – increase the size of segment “B”.

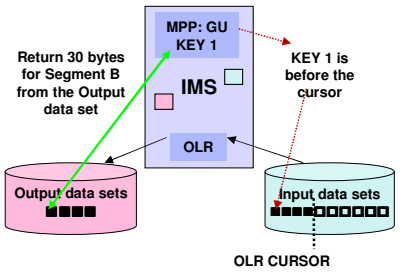
HALDB Alter Example (cont'd)

- As part of building the input and output DMB control blocks, IMS compares the segments and fields and creates a table of the deltas
- When ALTER is in progress, an application program reads "MASTER" looking for segment "B" with KEY1
  - If the segment key is before the OLR cursor
    - IMS reads segment B from the output data set whose size is 40 bytes
    - IMS checks the delta table before returning the segment to the application
    - Based on the table, IMS **returns 30 bytes** of data to the application

Delta Table

Name	Input	Output
B	30	40

Until ALTER and OLC are complete, IMS reads the segment using the old DMB and returns the unaltered segment structure.



Walkthru example showing how to make a structural change to the MASTER DB – increase the size of segment "B".

Until OLR ALTER and OLC are complete, IMS reads the segment using the old DMB and returns the unaltered segment structure

HALDB Alter Segment Field Fill Values

- For fields added at the end of a segment
  - If DBD field definition is TYPE ‘X’
    - Fill with x’00’
  - If DBD field definition is TYPE ‘P’
    - Fill low order byte with x’0C’ and other bytes with x’00’
  - If DBD field definition is TYPE ‘C’
    - Fill with x’40’
- When space without field definition(s) added to a segment
  - Fill with x’00’

Segment fill values are based on the field type:

X -> x’00’

P -> x’00…0C’

C -> x’40’

New added space with no fields(s) defined -> x’00’

HALDB Alter Operational Considerations

- All IMS data sharing systems must be running IMS 13
  - DBRC MINVERS value of “13.1” required
- Type-2 command environment required to initiate ALTER
  - Common Service Layer (CSL)
    - Structured Call Interface (SCI)
    - Operations Manager (OM)
  - Type-1 /INITIATE OLREORG command is not supported
- OLR processing is done for all partitions in a HALDB database
- Member OLC is recommended to bring specific ACBLIB(s) online
- Combine HALDB ALTER with new DB versioning

08- IMS 13 DB & DBRC: 563

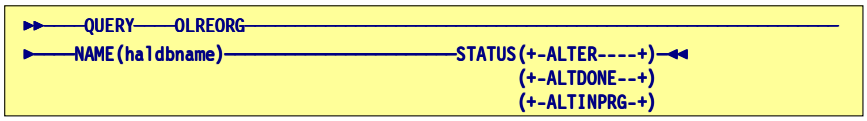
HALDB Alter Operational Considerations

- Enhanced QUERY OLREORG command
- New OLR return / reason codes, and completion codes for ALTER
- New “SF” status code
- New DFS and DSP messages
  - DFS1849E DFS3197I DFS3198I DFS3436E DFS3547E
  - DSP0174E DSP0175E DSP1097E
- Changed DFS messages
  - DFS047A DFS2991E
- Several new DBRC commands & modified LIST outputs

08- IMS 13 DB & DBRC: 564

HALDB Alter Type-2 QUERY Command

- QUERY OLREORG
  - Use QUERY to check on ALTER processing for a HALDB database



Keyword	Function
NAME() *optional*	• NAME(*) is the default, to query all defined HALDB partitions • specifies either one or more partition names or the name of a HALDB master • wildcard character (*) is not allowed, except as NAME(*)
STATUS()	• allows you to display the online reorganizations that possess a specified status
ALTER ALTDONE ALTINPRG	• specifies type of output for the partitions of the HALDB master, identified on the NAME() parameter: - status of processing for all partitions of a HALDB being altered - partitions for which alter processing is completed - partitions currently undergoing a structural change

The QUERY OLREORG command is used to query if a HALDB is undergoing a structural change process. To query information about alter processing, you can specify either one or more partition names or the name of a HALDB master.

The following keyword parameters are changed/added to the type-2 QUERY OLREORG command:

NAME() - This keyword specifies the name of a HALDB master or the name(s) of the HALDB PHDAM or PHIDAM partition(s) to be queried. NAME() is optional. A parameter with the wildcard character (*) is not allowed, except as NAME(*) for all defined HALDB partitions. NAME(*) is the default.

STATUS() - This keyword allows you to display the online reorganizations that possess the specified status.

ALTER - Displays status of alter processing for all partitions in a HALDB database that is being altered online

ALTDONE - Displays all partitions for which alter processing is complete

ALTINPRG - Displays all partitions that are currently being altered.

HALDB Alter Return, Reason and Completion Codes

- INITIATE OLREORG Return and Reason Codes

Return Code	Reason Code	Meaning
X'00000000'	X'00000000'	Command completed successfully.
X'00000010'	X'00004520'	Another OLR Alter in progress.

- INITIATE OLREORG Completion Codes

Completion Code	Completion Code Text	Meaning
0		Command completed successfully
1E1	OLR ITASK creation failed	OLR internal ITASK can not be created
1E2	Incorrect HALDB version detected	The ddir version for the database is different than the number recorded in the RECON DB record for the HALDB
1E3	Partition queued for OLR	The partition is being queued for HALDB alter processing
1E4	HALDB alter pending for Online Change	An alter request against the same HALDB was done, but an Online Change has not yet done for that HALDB
1E5	No DB structure change detected	An alter request is made but there are no database structure changes made
1E6	Insufficient CI/Block size detected	During alter processing, the CI/Block size of the database data set is smaller than the database largest segment size
1E7	Unsupported DBD changes detected	An alter request is made but the type of DBD changes made are not supported
(many more)	...	...
1EC	Logical database error	During alter processing, an error related to logical relationship on the altered DBD is detected

There are new return, reason and completion codes for the INIT OLREORG OPTION(ALTER) command.

HALDB Alter New "SF" Status Code

- Explanation
  - An application program that has field-level sensitivity attempted to read a database segment that was altered in length by the ALTER option of the HALDB online reorganization function.
    - Until the ACB members for the altered database are activated by the online change function, the altered segment cannot be accessed by application programs that have field-level sensitivity.
- System action
  - IMS returns this status code and continues to run normally
- System programmer response
  - Use the member online change function to activate the ACB members for the altered database and rerun the application program

New SF Status Code: programs with field-level sensitivity cannot access segments with altered length until the OLC process is completed to bring the new ACB member (s) online.

HALDB Alter DBRC LIST Command Output

- LIST output for a HALDB Master Database contains new info
  - "ALTER COUNT" indicates number of partitions that are to be altered
  - "ALTER COMPLETE COUNT" indicates number of partitions for which the alter process has completed

```
DB
 DBD=DBOHIDK5                          DMB#=3      CHANGE#=3      TYPE=HALDB
 SHARE LEVEL=3                 GSGNAME=**NULL**
 DBRCVGRP=**NULL**
 PSNAME=**NULL**  DBORG=PHIDAM    DSORG=OSAM    CURRENT PARTITION ID=00001
 FLAGS:                              COUNTERS:
   RECOVERABLE           =YES          PARTITIONS               =4
   ONLINE REORG CAPABLE  =YES          DATA SET GROUP MEMBERS   =2
                                       ALTER COUNT              =4
                                       ALTER COMPLETE COUNT     =2
```

-Number of partitions to be altered
-Number of partitions altered

There is additional information stored in the DB and Partition RECON records about HALDB ALTER processing.

HALDB Alter DBRC LIST Command Output (cont'd)

- LIST output for a HALDB Database Partition contains new info
  - "ALTER BLOCK SIZE" lists new block sizes to be used by the alter process
  - "ALTER IN PROGRESS" indicates whether the alter process has started
  - "PARTITION ALTERED" indicates whether the alter process has completed

only for OSAM

```
OSAM BLOCK SIZE:
  A = 4096
  B = 4096

ALTER BLOCK SIZE:
  A = 0
  B = 8192
```

'0' indicates no BLKSIZE change

```
FLAGS:
     BACKOUT NEEDED          =OFF
     READ ONLY               =OFF
     PROHIBIT AUTHORIZATION=OFF

     TRACKING SUSPENDED      =NO
     OFR REQUIRED            =NO
     PARTITION INIT NEEDED   =NO
     OLREORG CURSOR ACTIVE   =NO
     PARTITION DISABLED      =NO
     ONLINE REORG CAPABLE    =YES
     REORG INTENT            =NO
     QUIESCE IN PROGRESS     =NO
     QUIESCE HELD            =NO
     ALTER IN PROGRESS       =NO
     PARTITION ALTERED       =NO
```

```
COUNTERS:
     RECOVERY NEEDED COUNT    =0
     IMAGE COPY NEEDED COUNT  =0
     AUTHORIZED SUBSYSTEMS    =1
     HELD AUTHORIZATION STATE=3
     EEQE COUNT               =0
     RECEIVE REQUIRED COUNT   =0
     OLR ACTIVE HARD COUNT    =0
     OLR INACTIVE HARD COUNT  =0
```

Partition values

There is additional information stored in the DB and Partition RECON records about HALDB ALTER processing.

HALDB Alter DBRC LIST Command Output (cont'd)

```
DB
  DBD=POHIDKA   MASTER DB=DBOHIDK5  IRLMID=*NULL  CHANGE#=3     TYPE=PART
  USID=0000000002  AUTHORIZED USID=0000000002  HARD USID=0000000002
  RECEIVE USID=0000000002  RECEIVE NEEDED USID=0000000000
  DSN PREFIX=IMSTESTS.DBOHIDK5                          PARTITION ID=00001
  PREVIOUS PARTITION=**NULL**    NEXT PARTITION=**NULL**
  OLRIMSID=**NULL**    ACTIVE DBDS=M-V
  REORG#=00000
  ONLINE REORG STATISTICS:
      OLR BYTES MOVED = 5576000
      OLR SEGMENTS MOVED = 16000
      OLR ROOT SEGMENTS MOVED = 4000

  FREE SPACE:
    FREE BLOCK FREQ FACTOR=0    FREE SPACE PERCENTAGE=50

  PARTITION HIGH KEY/STRING (CHAR):               (LENGTH=5  )
    .....
  PARTITION HIGH KEY/STRING (HEX):
    FFFFFFFFFF

  OSAM BLOCK SIZE:
    A = 4096
    B = 4096
  ALTER BLOCK SIZE:
    A = 0
    B = 8192

  FLAGS:                                COUNTERS:
    BACKOUT NEEDED         =OFF         RECOVERY NEEDED COUNT   =0
    READ ONLY              =OFF         IMAGE COPY NEEDED COUNT =0
    PROHIBIT AUTHORIZATION=OFF          AUTHORIZED SUBSYSTEMS   =1
                                        HELD AUTHORIZATION STATE=3
                                        EEQE COUNT              =0
    TRACKING SUSPENDED     =NO          RECEIVE REQUIRED COUNT  =0
    OFR REQUIRED           =NO          OLR ACTIVE HARD COUNT   =0
    PARTITION INIT NEEDED =NO           OLR INACTIVE HARD COUNT =0
    OLREORG CURSOR ACTIVE =NO
    PARTITION DISABLED     =NO
    ONLINE REORG CAPABLE   =YES
    REORG INTENT           =NO
    QUIESCE IN PROGRESS    =NO
    QUIESCE HELD           =NO
    ALTER IN PROGRESS      =NO
    PARTITION ALTERED      =NO
```


Sample listing of DBRC Partition record

HALDB Alter DBRC LIST Command Output (cont'd)

- LIST output for a REORG contains new info
  - REORG record now indicates whether the HALDB partition DBDS was altered during the online reorganization

```
REORG
   RUN      = 12.063 12:19:55.476258          *    USID = 0000000002
   REORG#   = 00005                                        ALTER
   STOP     = 12.063 12:20:13.210119            ONLINE   RECOV = NO
```

Partition was altered

DBRC REORG record contains a new indicator showing whether ALTER was part of the OLR for this partition.

HALDB Alter DBRC API Query Output

- API query request output has changed
  - DSPAPQHP – HALDB Partition output block
    - Now returns the array of new Block / CI sizes to be used by the OLR alter process
    - Will only exist if the alter process is still in progress
  - DSPAPQHB – HALDB output block
    - Two new counters:
      - the total number of partitions being altered
      - the number of partitions for which the alter process has completed
  - DSPAPQRR – DBDS reorganization output block
    - New flag which indicates whether an online reorganization altered the HALDB

Some of the DBRC API query output has changed. There are several new fields that are returned when the OLR ALTER function is being used.

HALDB Alter DBRC RECON Records

- Content of several DBRC records have changed
  - DSPDBHRC – Database record (DB)
    - no size increase
  - DSPPTNRC – Partition record (PART)
    - size increases only when ALTER is running
  - DSPRRGRC – Reorganization record (REORG)
    - no size increase

Remember to reassemble any programs using the changed DBRC control blocks !

Some of the DBRC records have changed in support of the OLR ALTER process.

HALDB Alter DBRC Command Changes

- CHANGE.DB
  - **ALTER | NOALTER**
    - New, optional keywords
    - Specifies whether the HALDB partition is in the process of being altered.
    - Cannot be changed if the partition is authorized
- NOTIFY.REORG
  - **ALTER**
    - New, optional keyword
    - Specifies that OLR **altered** the database structure
    - Indicates that OLR **is altering** the database structure
    - ALTER can only be specified with ONLINE

Only use these commands in cases of a failure where RECONs need to be cleaned up !

These DBRC commands are used internally by IMS to set flags for ALTER processing.

IMS Users should not have to use these commands unless required to cleanup RECON records after a major system failure.

HALDB Alter Security Considerations

- Security Considerations
  - New ALTER form of Online Reorg can change a database structure
    - INITIATE OLREORG command should be secured
  - RACF attributes for Type-2 INIT command

IMS Command	Command Keyword	RACF access authority	Resource name
INIT	OLREORG	UPDATE	IMS.plxname.INIT.OLREORG

Be sure you have the INIT OLR command secured.

HALDB Alter Performance Considerations

- Performance Characteristics
  - Reorganizing a HALDB with the ALTER option
    - Performance should be same as reorganizing a HALDB without the ALTER option

Overall elapsed time to complete an INIT OLREORG OPTION(ALTER) command for a HALDB database is likely to be significantly greater than an INIT OLREORG for a single HALDB database partition or even a subset of partitions. Remember – the ALTER option applies to **ALL** partitions in a HALDB database in one single pass.

If BLKSIZE for a PHDAM database changes, IMS will take an extra lock per RAP during ALTER processing.

HALDB Alter Summary

- IMS 13 provides ability to make structural changes to a HALDB database without a database outage
  - Structural changes can be made to DB segment definitions
  - Online Reorganization is used to apply the structural changes to the online HALDB database
  - Online Change process is used to activate the new ACBLIB member(s) in the online IMS system
- Benefit
  - Eliminate a database outage when structural changes must be made to segment definitions in a DBD
  - Improved online availability of HALDB databases

HALDB Alter is for HALDB database customers who want to have the ability to make segment changes without unloading and reloading the database.

This addresses the challenge of maintaining database availability while changing the structure of a HALDB database. The actual hierarchy of the IMS HALDB cannot be changed.

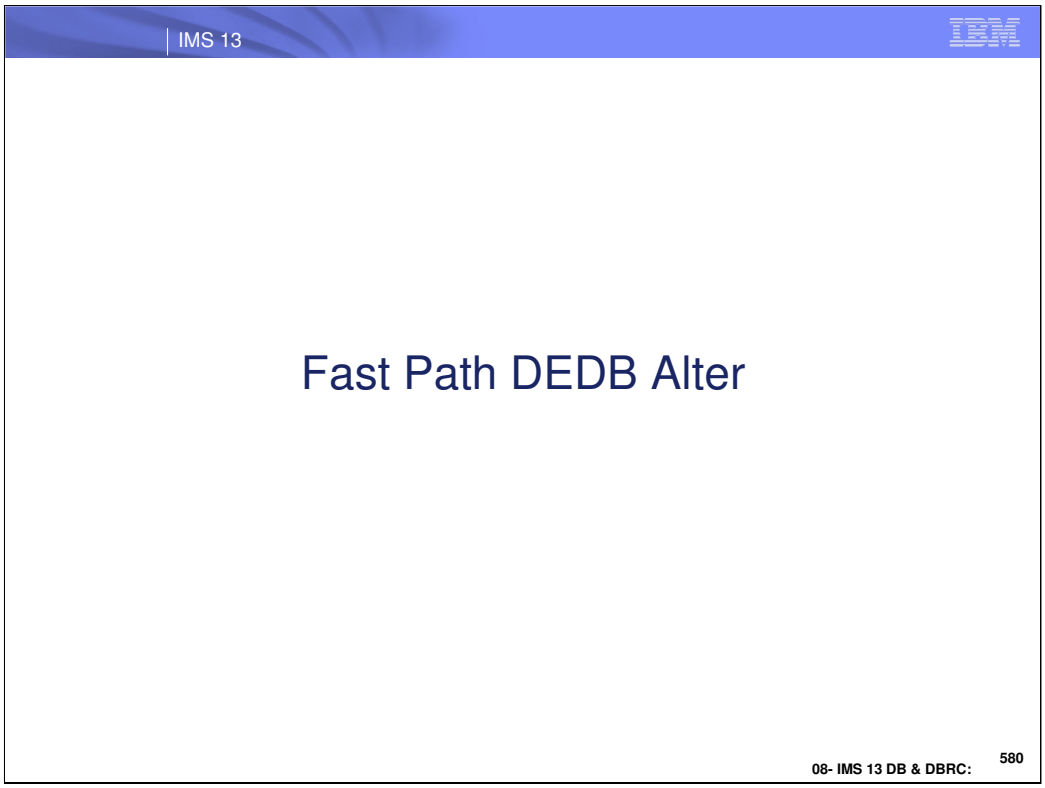
When a segment change is made to a DBD, an online command can be issued to apply the change to the database. The change is implemented via an option of the HALDB Online Reorganization (OLR) function. Application programs can access the database at the same time the OLR function is changing the structure of the database.

This line item provides value to customers by reducing the complexity of making structural changes to a HALDB database, eliminating system down time, and improving system availability. This line item also reduces the cost and risk associated with making and coordinating wholesale changes to all application programs when database structure changes occur. This line item allows customers to improve their HALDB database structures even if the database is used by critical application programs that they no longer have the ability to change.

Fast Path Enhancements

This section describes the changes in the Fast Path area for IMS 13.

There are two new features of IMS 13. The first is DEDB Alter. The second is a set of PTFs for Secondary Index Enhancements that are being forwarded fitted into IMS 13.

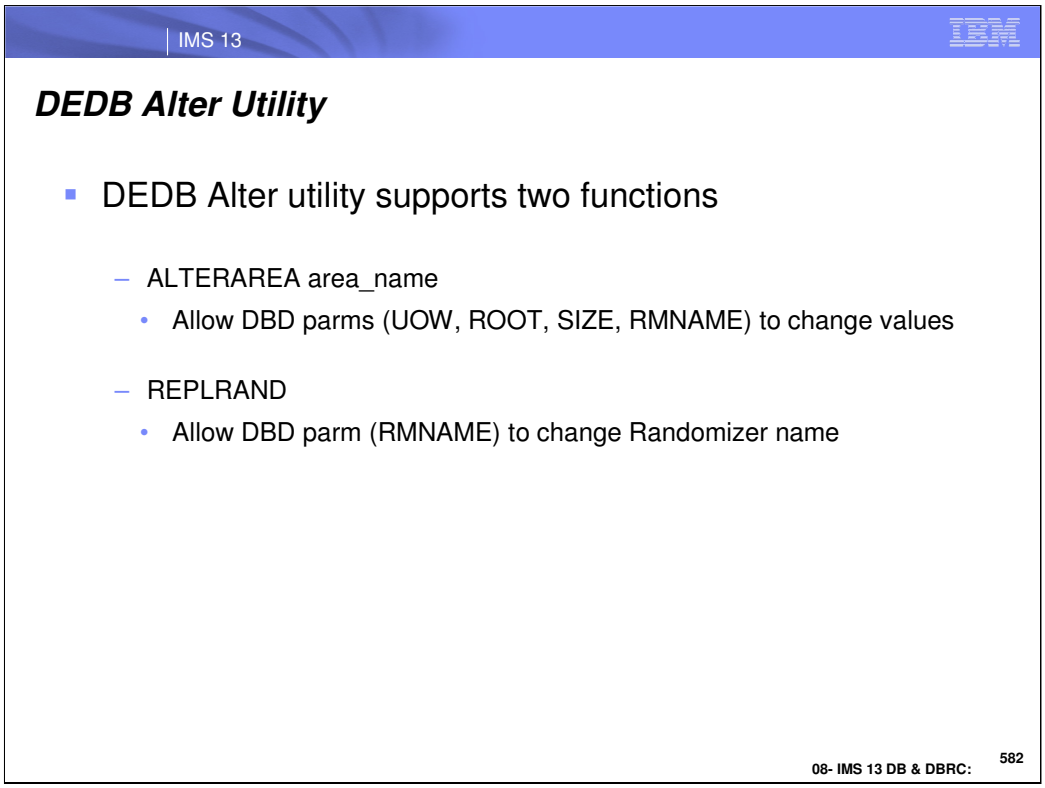


This section addresses the DEDB Alter function in IMS 13.

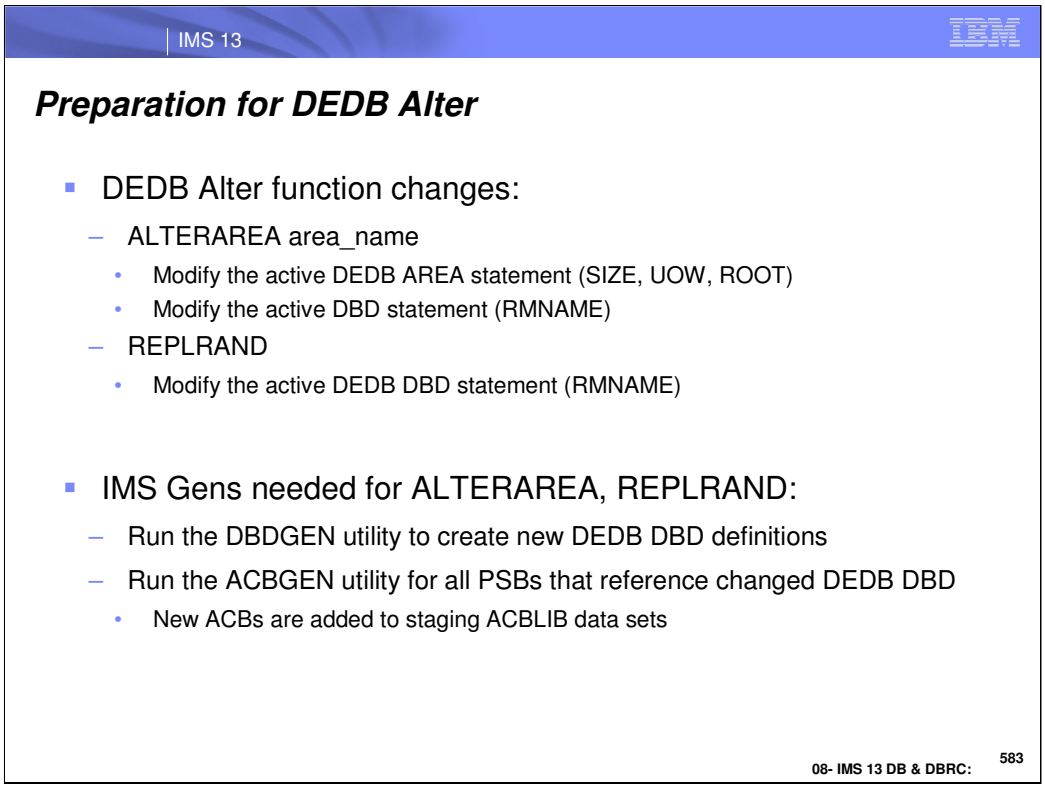
DEDB Alter

- IMS 13 adds ability to dynamically change DEDB specifications
  - Users can dynamically change UOW, SIZE, ROOT, Randomizer while DEDB is online
  - New DEDB Alter utility is used for DEDB changes
  - DRD is not required for DEDB Alter
  - Supports VSO Areas if /VUNLOAD is done before DEDB Alter is executed
- Benefits
  - Improved management of DEDB definitions
    - Eliminate system down time for modifications to DEDB definitions
    - Improve data availability since changes are done while DEDB is online

In IMS 13, Fast Path has added the ability to dynamically change specific DEDB specifications. For example, the UOW, SIZE, ROOT, and Randomizer routine can be changed while the DEDB is online. There is a new DEDB Alter utility that allows these DEDB changes to occur. This is support is available for VSO areas provided the areas are unloaded first using the /VUNLOAD command. Allowing dynamic changes to the DEDBs will improve data availability and reduce system down time.

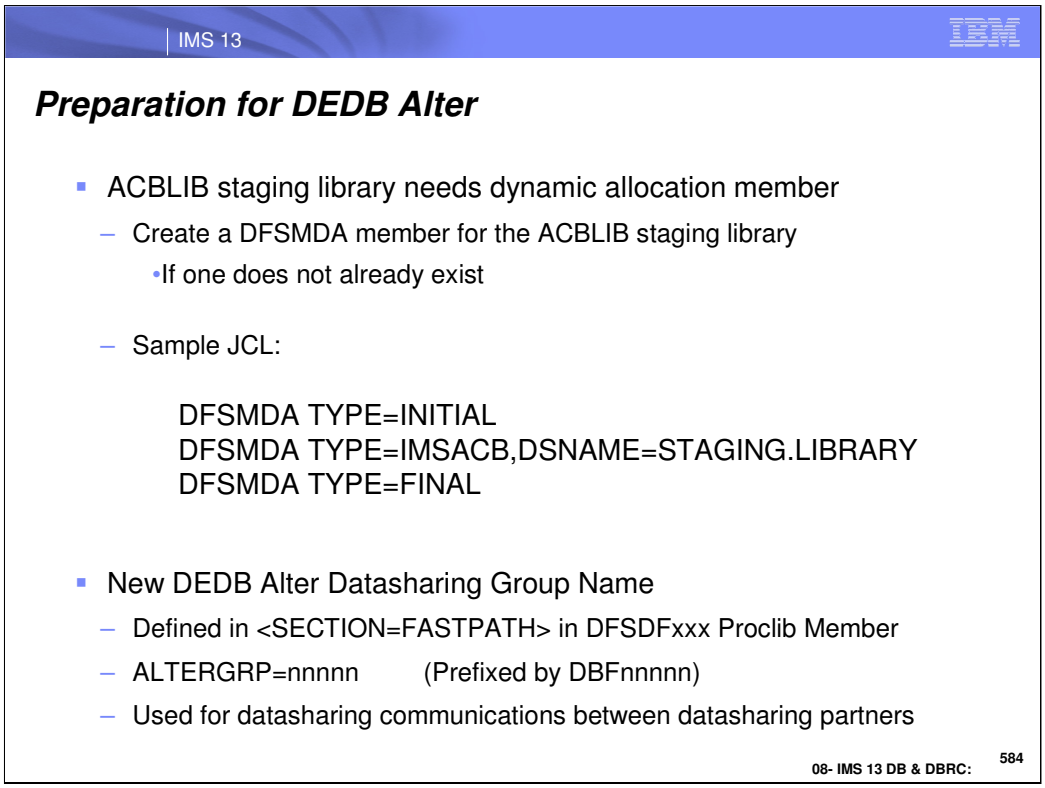


The DEDB Alter utility supports two new functions. ALTERAREA is used to change the UOW, ROOT, SIZE, and RMNAME (randomizer name) values. REPLRAND is used to specifically change the Randomizer name.



There is some preparation needed before the DEDB Alter utility can make the DEDB changes. For the ALTERAREA function, the active DEDB AREA statement must be modified with the new SIZE, UOR, or ROOT definitions. If the RMNAME parameter is used, the new randomizer name must be different than the existing randomizer name. For the REPLRAND function, the new randomizer must be assembled and link edited into the IMS SDFSRESL STEPLIB concatenation. The active DEDB AREA statement must be modified. The active randomizer must be a 2-stage randomizer and the new active randomizer must also be a 2-stage DEDB randomizer.

After the DEDB AREA statements are modified, the DBDGEN utility is run to create new DEDB DBD definitions. The ACBGEN utility is run next for all PSBs that reference the changed DEDB DBD. The new ACBs resulting from the ACBGEN are added to the staging ACBLIB data sets.



The ACBLIB staging library needs to have a dynamic allocation member. A DFSMDA member can be created for the ACBLIB staging library if one doesn't already exist using the above DFSMDA macros.

There is a new parameter in the <SECTION=FASTPATH> area of the DFSDFxxx proclib member called ALTERGRP=nnnnn. This paramter allows the user to define a new datasharing group name (DBRnnnnn) which allows datasharing partners to communicate with each other during DEDB Alter utility execution. There is no default for this parameter.

Preparation for DEDB Alter

- DEDB Alter commits new DBD from Staging ACBLIB Library into Active ACBLIB
  - For IMS datasharing, need to know if:
    - Each IMS is ***sharing same*** ACBLIB (ACBSHR=Y)
    - Each IMS has ***its own*** ACBLIB (ACBSHR=N)
  - If Common Service Layer is ***not used***
    - ACBSHR=Y|N is under <SECTION=FASTPATH> in DFSDFxxx
  - If Common Service Layer ***is used***
    - ACBSHR=Y|N uses following precedence:
    - 1st DFSCGxxx PROCLIB member
    - 2nd <SECTION=COMMON_SERVICE_LAYER> in DFSDFxxx
    - 3rd <SECTION=FASTPATH> in DFSDFxxx
  - DEDB Alter uses ACBSHR setting from local IMS system
    - All IMS datasharing systems must have same ACBSHR setting

Once the ACB is placed into the staging ACBLIB, DEDB Alter commits the new DBD and moves it into the Active ACBLIB. In a datasharing environment, the IMS system can share the ACBLIB with the other IMS subsystems in the datasharing environment or each IMS can have its own ACBLIB. The ACBSHR=Y|N parameter indicates how the ACBLIB is used between the IMS subsystems. The ACBSHR specification is found in the FASTPATH section in the DFSDFxxx proclib member if the Common Service Layer (CSL) is used in the environment. If CSL is not used, then the ACBSHR specification is found using the following precedence. The first check is in the DFSCGxx proclib member. The second check is in the COMMON_SERVICE_LAYER section in the DFSDFxxx proclib member. The third check is in the FASTPATH section in the DFSDFxxx proclib member. While DEDB Alter uses the ACBSHR setting that is found in the local IMS subsystem, all IMS subsystems in the datasharing environment must have the same ACBSHR parameter setting.

Preparation for DEDB Alter

- Allocate Shadow Area data sets
  - Shadow Area data sets used for migrating existing data from Active Areas
  - Single Area Data Sets (SADS)
    - Single Shadow data set is required
  - Multiple Area Data Sets (MADS)
    - 2 to 7 Shadow data sets are required
  - SADS can become MADS and MADS can become SADS after DEDB Alter
    - Depends on the number of allocated Shadow Area data sets
  - Shadow data sets are only for DEDB Alter utility
- Allocate Shadow IC data sets:
  - Created while DEDB Alter migrates data to Shadow Area data sets
- Shadow Area data sets + Shadow Image Copy data sets <= 7

The next step in the preparation for DEDB Alter is to allocate both the Shadow Area data sets and the Shadow Image Copy data sets. The Shadow Area data sets are used for migrating the existing data from the Active Areas to the Shadow Areas. The Shadow Area data set can be a Single Area Data Set (SADS) or it can be a Multiple Area Data Set (MADS) if there are two to seven Shadow Area data sets allocated. It is also possible to turn a SADS into a MADS after the DEDB Alter is run by allocating additional Shadow Area data sets. The Shadow Area data sets are used exclusively by the DEDB Alter utility and are not accessible by the IMS subsystem. The Shadow Area Image Copy data sets are created during the migration of data to the Shadow Area data sets.

Preparation for DEDB Alter

- Register Shadow data sets to DBRC (INIT.ADS)

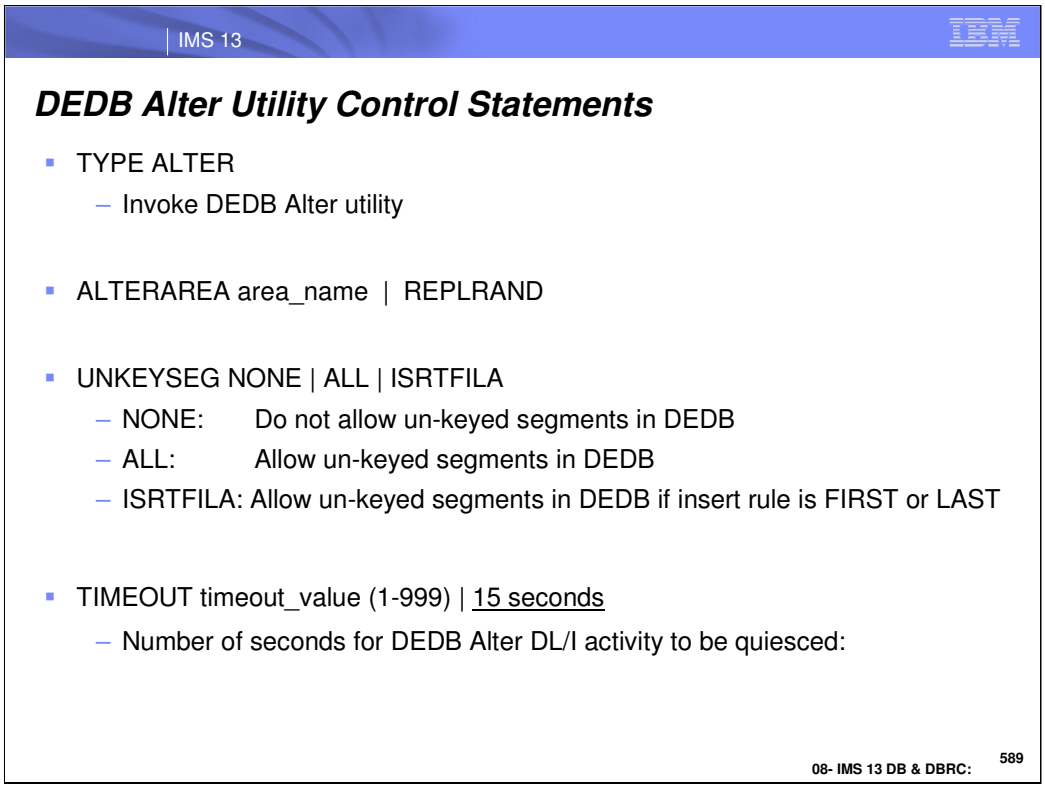
```
>>-INIT.ADS--ADDN(shadowname)--ADSN(shadowname)--AREA(name)--DBD(name)--->

                  .-UNAVAIL-----.
               >--+---------+-------+--------------+------------------------><
                  '-AVAIL----------'                '-SHADOW------+-------'
                                                                  '-IC----'
```


Prior to executing the DEDB Alter utility, the Shadow Area and Shadow Image Copy data sets must be allocated. Once allocated, they can be registered to DBRC using the INIT.ADS command. The DEDB Area Initialization utility (DBFUMIN0) formats the Shadow Area and Shadow Area Image Copy data sets and flags them as available in the RECON data set. There must be at least one Shadow Area and Shadow Area Image Copy data sets flagged as Available in the Recon.

Preparation for DEDB Alter

- Format Shadow data sets with DEDB Area Init Util (DBFUMIN0)
  - Shadow Area data sets and Shadow IC data sets formatted
  - Two new control cards: ACTIVE | SHADOW
    - ACTIVE = Format Area data sets for DEDB Area
    - SHADOW = Format Shadow Area and Shadow IC data sets
  - Formats Active or Shadow data sets in one execution, but not both
  - When DBRC=Y, formats both Shadow Area and Shadow IC in one execution
- New ACB from Staging ACBLIB used for formatting
- After utility completes, flags are set in RECON:
  - Shadow Area data sets are marked “SHADOW AVAIL”
  - Shadow IC Area data sets are marked “SHADOW IC AVAIL”
- Shadow data sets must be formatted before DEDB Alter utility runs

The DEDB Area Initialization Utility (DBFUMIN0) has been enhanced to format the Shadow Area data sets and the Shadow Area Image Copy data sets. The ACTIVE keyword indicates that the DEDB Area data sets are to be formatted. The SHADOW keyword indicates that the Shadow Area and Shadow Area Image Copy data sets are to be formatted. This utility will format either the Active DEDB Area data sets or the Shadow Area and Shadow Area Image Copy data sets in one execution, but it can not do both types of data sets in one execution. If DBRC=Y, then it can format both the Shadow Area data sets and the Shadow Area Image Copy data sets in one execution. The Shadow Area and Shadow Area Image Copy data sets can be formatted while the Active DEDB Area data sets are online. After the utility completes, there are flags set in the DBRC Recon data set. The Shadow Area data sets are flagged as “SHADOW AVAIL” and the Shadow Area Image Copy data sets are flagged as “SHADOW IC AVAIL”. The Shadow Area and Shadow Area Image Copy data sets both must be formatted before the DEDB Alter utility can run.



The DEDB Alter utility uses the following control cards for execution. The TYPE ALTER invokes the DEDB Alter utility. There are two functions and they are: 1) ALTERAREA and 2) REPLRAND. The UNKEYSEG keyword determines whether unkeyed segments are allowed in the DEDB. The TIMEOUT value indicates how long it can take for the DEDB Alter to quiesce the DL/I activity when suspending IMS applications.

DEDB Alter Utility Control Statements (continued)

- RETRY NO | YES | retry_value (1-99)
  - NO: Do not retry after TIMEOUT value expires.
  - YES: Retry after TIMEOUT value until the utility completes successfully
  - retry_value: Number of utility retries after TIMEOUT value occurs
- RETRYWAIT retrywait_value (1-999) | 60
  - If RETRY YES or RETRY retry_value
    - Number of seconds to wait before retrying the commit process
- GO
  - Execute the DEDB Alter utility

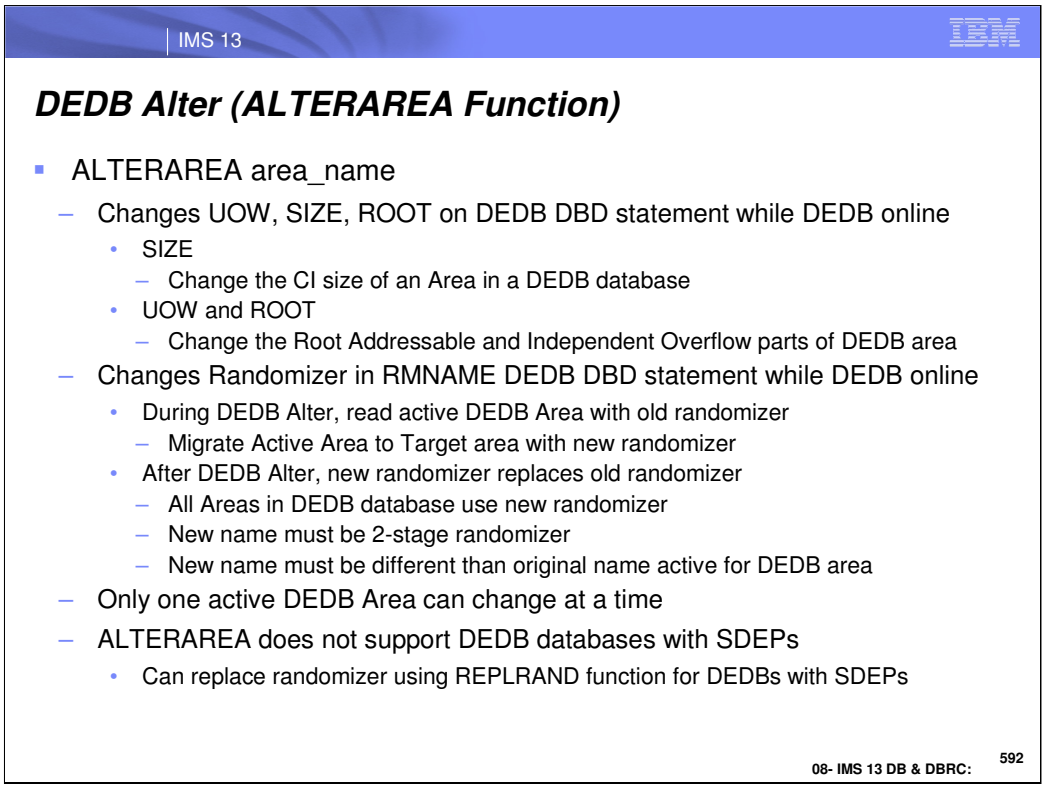
The RETRY keyword indicates whether to retry the DEDB Alter function if a timeout occurs. A specification of “NO” indicates there should be no retried after the timeout occurs. A specification of “YES” indicates the retries should continue until the utility is successful. A specification of “retry_value” indicates the number of retries that can be attempted after the timeout occurs. The RETRYWAIT keyword indicates the number of seconds to wait before the next retry of the commit process. Finally, the GO keyword initiates the DEDB Alter utility execution.

DEDB Alter Utility Execution (ALTERAREA)

- Sample JCL: DEDB Alter Utility (ALTERAREA)

```
//ALTAREA  JOB ….
//FPUTIL   PROC  SOUT=A,RGN=1M,
//               DBD=,REST=00,DIRCA=002,
//               PRLD=,IMSID=,AGN=,SSM=,ALTID=
//FPU      EXEC  PGM=DFSRRC00,REGION=&RGN,
//               PARM=(IFP,&DBD,DBF#FPU0,&REST,00,,1,
//               &DIRCA,&PRLD,0,,,,&IMSID,&AGN,&SSM,,
//               &ALTID)
//STEPLIB  DD DSN=IMS.CRESLIB,DISP=SHR
//PROCLIB  DD DSN=IMSVS.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT,…
//S0          EXEC FPUTIL,RGN=1M,DBD=DEDBJN21,REST=00,IMSID=IMS1
//SYSIN       DD   *
TYPE          ALTER
ALTERAREA     DB21AR0
RETRY         NO
TIMEOUT       30
GO
/*
```


This is sample JCL showing the DEDB Alter execution control cards for the ALTERAREA function.

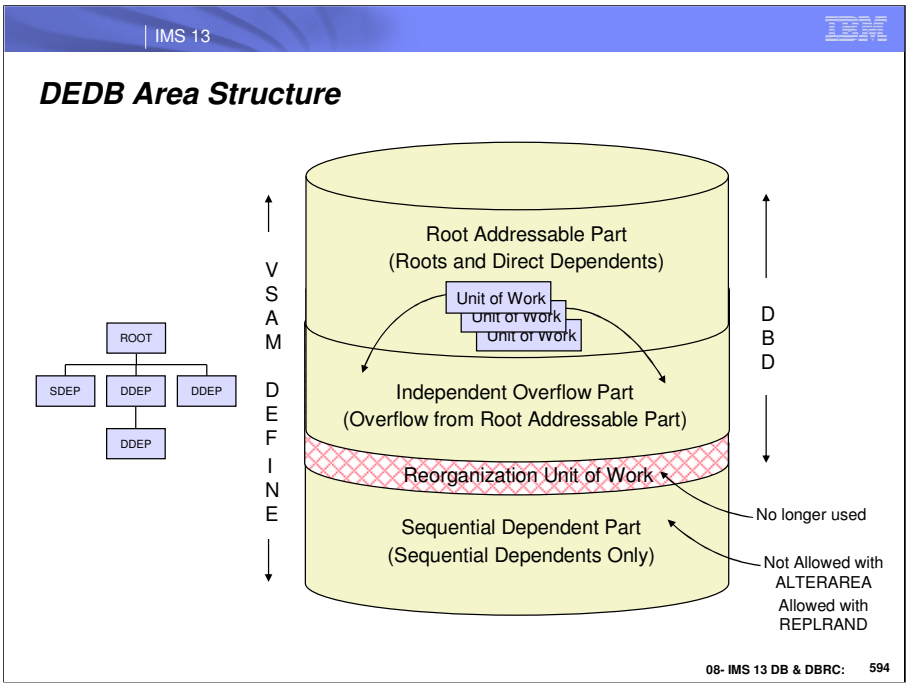


The DEDB Alter utility ALTERAREA function changes the UOW, SIZE, ROOT values while the DEDB Area data set is online. The SIZE parameter affects the CI size of an Area. The UOW and ROOT parameters affect the Root Addressable and Independent Overflow parts of the DEDB Area. The RMNAME parameter changes the Randomizer name while the DEDB Area data set is online. When a Randomizer name is being change, the Active DEDB Area data sets are read using the existing randomizer routine. The data is migrated from the Active DEDB Area data sets to the Target DEDB Area data sets using the new Randomizer name. After the DEDB Alter is completed, the new randomizer name replaces the existing randomizer name. At that point, all DEDB Areas start to use the new randomizer name. The new name must be a two-stage randomizer and it must be a different name from the existing randomizer name. There can be only one Active DEDB Area Data set changed at a time. It is not possible to run the DEDB Alter utility concurrently for another DEDB Area in the same DEDB database. The ALTERAREA function does not support changes to DEDB databases with SDEPs. However, it is possible to replace the randomizer name using the REPLRAND function for DEDBs with SDEPs.

Relationship of UOW, ROOT and SIZE Parameters

- UOW=(number1,overflow1)
  - Number of Control Intervals (CI) in a UOW
    - Number1 = number of Control Intervals (CI) in a UOW
    - Overflow1 = number of Control Intervals in overflow section of UOW
- ROOT=(number2,overflow2)
  - Space allocated to Root Addressable Part and Independent Overflow
    - Number2 = Space (in UOWs) for Root Addressable Part and Independent Overflow
    - Overflow2 = Space (in UOWs) for Independent Overflow
- SIZE=value
  - Control Interval (CI) size (in Bytes)
    - 512 bytes, 1 KB, 2KB, 4KB, 8KB, 16KB, 20KB, 24KB, 28KB
    - SIZE value must match the CI size defined to VSAM

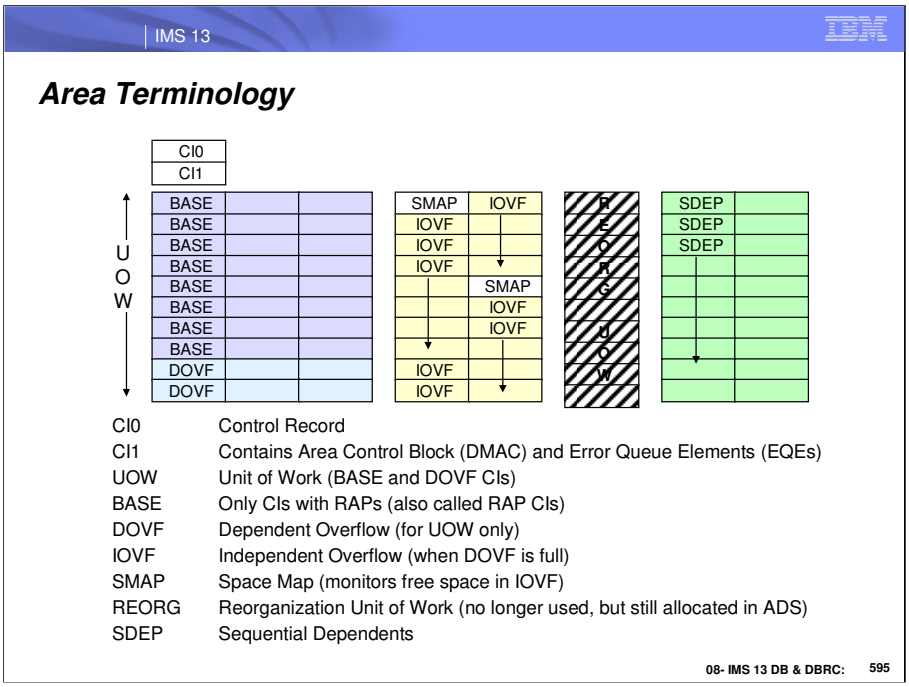
The UOW, ROOT, and SIZE parameters can be changed by the DEDB Alter utility. The UOW parameter has two values and determines the number of Control Intervals (CI) in a UOW and the number of CI in the overflow section of the UOW. The ROOT parameter has two values and determines the space allocated to the Root Addressable Part and the Independent Overflow. The SIZE parameter indicates the size of the CI in bytes.



The "direct part" of a DEDB area consists of the root addressable part and the independent overflow part. These parts may contain both root segments and their direct dependents, but not sequential dependent segments. The root addressable part is further divided into groups of CIs called "Units of Work" or UOWs. The independent overflow part holds roots and DDEPs which have overflowed from the direct part, similar to (but not quite the same as) the way overflow works for HDAM.

Following the direct part is a group of CIs called the "reorganization unit of work." These CIs are no longer used by Fast Path, but are still there for compatibility reasons.

Whatever space is left over from the direct part and reorg UOW is used for SDEPs. There are no parameters in the DBD to define how much space is to be used for SDEPs - it is just the difference between the VSAM DEFINE and what is used by the other parts.



This diagram shows some terminology that applies to a DEDB area.

The first two CIs in an ADS contain control information and a control block called the DMAC. CI0 is not very interesting at all, but we will talk about CI1 and the DMAC quite a bit.

The UOW mentioned on the previous visual consists of two types of CIs - BASE (sometimes called RAP) CIs and DOVF (dependent overflow) CIs. The DOVF CIs are used only for the overflow of BASE CIs in the same UOW.

Independent overflow also contains two types of CIs - Space Map (SMAP) CIs which have a similar function to the bit maps in HDAM or HIDAM databases, and the IOVF CIs themselves, which contain roots and DDEPs which have overflowed from the UOWs and their DOVF CIs,

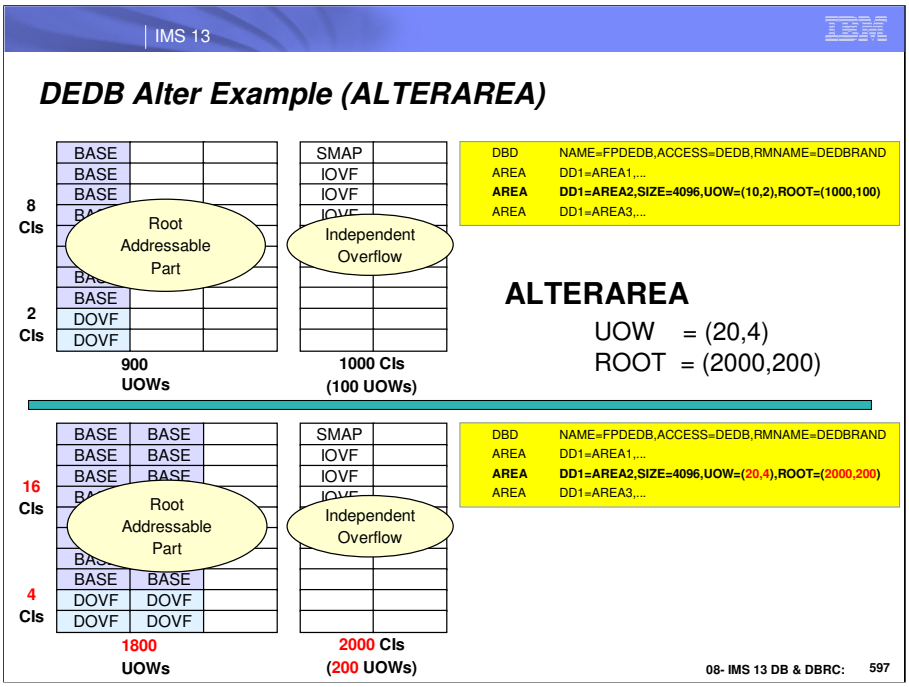
As mentioned before, the REORG UOW is no longer used by Fast Path, but still exists in the data set for compatibility reasons (wouldn't want to have to reorg every area just to get rid of them).

And finally the SDEP CIs, which are all the CIs from the end of the REORG UOW to the end of the VSAM ESDS.

Examples of Increasing DEDB Area size

- Expand the DEDB Area using ***same*** Control Interval (CI) size
  - Increase UOW= to put more CIs in a UOW
  - Increase ROOT= to allocate more space for Root Addressable Part and Independent Overflow
- Expand the DEDB Area using ***different*** Control Interval (CI) size
  - Increase SIZE= to create a larger CI size
  - Increase UOW= to put more CIs in a UOW
  - Increase ROOT= to allocate more space for Root Addressable Part and Independent Overflow

Here are two examples of how the size of the DEDB Area can be changed. In the first case, the DEDB Area size is increased without changing the CI size. Instead, the UOW parameter is changed to increase the number of CIs in a UOW. The ROOT parameter is also changed to allocate more space for the Root Addressable Part and the Independent Overflow. In the second case, the DEDB Area size is increased using a different CI size. In this example, the SIZE parameter uses a larger CI size, the UOW is increased to put more CIs in the UOW, and the ROOT parameter is increased to allocate more space for the Root Addressable Part and the Independent Overflow.



Each area in a DEDB is defined by an AREA statement, replacing the DATASET statement used for HDAM and HIDAM databases. The AREA statement defines the area name (or DD name if the area is not registered), the CI size, the size and configuration of a UOW, and how much independent overflow to allocate. The values of these parameters can be different for each AREA.

After all the AREA statements, the SEGM and FIELD statements define the hierarchical structure of the database. All areas have the same structure.

DEDB Alter Utility Execution (ALTERAREA)

- Complete preparation steps:
  - Alter DEDB DBD, run DBDGEN, run ACBGEN
  - Allocate Shadow Area and Shadow IC data sets, Register and Format
- Execute DEDB Alter Utility (ALTERAREA)

A. Data is migrated from active DEDB Area to Shadow Area and Shadow IC
- Uses new UOW, ROOT and SIZE parms in staging ACBLIB
- Use current randomizer to read Active DEDB Area
- Use new randomizer (if changing) to insert to Shadow Area data set

B. Commits new randomizer and/or UOW, ROOT and SIZE changes
1. Active DEDB Area is quiesced and DL/I calls are suspended
2. Shadow Area data set is synchronized with Active DEDB Area
3. Changed ACB in the Staging ACBLIB is moved to the Active ACBLIB
   - New randomizer replaces existing randomizer (if changed)
4. Shadow Area data set becomes new DEDB Area data set
   - Original DEDB Area data set is preserved
5. Un-Quiesce DEDB Area and resume suspended DL/I calls

After the DEDB DBD has been altered, the DBDGEN and the ACBGEN have been executed, the Shadow Area and Shadow Area Image Copy data sets have been allocated, registered and formatted, it is time to execute the DEDB Alter utility. The ALTERAREA function migrates the data from the Active DEDB Area data sets to the Shadow Area data sets and creates the Shadow Area Image Copy data sets. In the process, any changes to the UOW, ROOT, or SIZE parameters in the Staging ACBLIB are implemented. The current randomizer is used to read the Active DEDB Areas, but the new randomizer is used to insert to the Shadow Area data sets.

When the changes are committed, the Active DEDB Area data sets are quiesced and any DL/I calls are suspended. The Shadow Area data sets are synchronized with the Active DEDB Area data sets. The changed ACB in the Staging ACBLIB is moved to the Active ACBLIB and, if there is a new randomizer, it replaces the existing randomizer. The Shadow Area data set becomes the new DEDB Area data set preserving the original DEDB Area data set. Finally, the DEDB Area is un-quiesced resuming suspended DL/I calls.

Post-DEDB Alter Utility Execution

- If DEDB Alter is Successful
  - Shadow Area data set is promoted to Active Area data set
    - Old active area data set is demoted to SHADOW area data set
  - Shadow IC data set is promoted to user Image Copy
    - Registered in DBRC as “user image copy”
      - Counted as user image copy in GENMAX value
    - Not a standard image copy
      - Instead, it is an image of Active Area data set
    - To recover the Area with this image copy
      - (1) Notify DBRC that Area was restored
        - NOTIFY.RECOV DBD(name) AREA(name) RCVTIME(time_stamp)
        - time_stamp = Time when SHADOW IC was created by DEDB Alter utility
      - (2) Issue GENJCL.RECOV with no image copy
        - GENJCL.RECOV DBD(name) AREA(name) USEAREA

If the DEDB Alter function is successful, the Shadow Area data set is promoted to the Active Area data set replacing the previous Active Area data sets which become the Shadow Area data sets. The Shadow IC data set is promoted to a User Image Copy. In DBRC, this User Image Copy is registered as a User Image Copy so that it can be counted in the GENMAX count. It is note a Standard Image Copy, it is just an image of the Active Area data set. To recover a DEDB Area with this User Image Copy, the user must notify DBRC (NOTIFY.RECOV) with the name of this Area data set and the time when the Shadow Image Copy was created (i.e. RCVTIME). Specifying the GENJCL.RECOV command without an image copy name will pick up this Shadow Image Copy name.

Post-DEDB Alter Utility Execution

- If DEDB Alter is Unsuccessful
  - Active Area data sets remain active and accessible to IMS systems
  - If Shadow Area data sets are marked as AVAIL:
    - Shadow Area data sets have not been written to
    - Can be used in subsequent DEDB Alter utility executions
  - If Shadow Area data sets are marked as UNAVAIL:
    - Shadow Area data sets have been written to
    - Shadow Area data set must be:
      - Re-allocated
      - Re-formatted with DBFUMIN0

If the DEDB Alter was unsuccessful, the Active Area data sets remain active and are still accessible to the IMS systems. The flags in DBRC are checked to see if any data was written to the Shadow Area data sets. If the flag still shows AVAIL, then no data was written to the Shadow Area data sets and they can be used in subsequent DEDB Alter utility executions. If the flag shows UNAVAIL, then data was written to the Shadow Area data sets and they need to be re-allocated and re-formatted.

DEDB Alter Utility Execution (REPLRAND)

- Sample JCL: DEDB Alter Utility (REPLRAND)

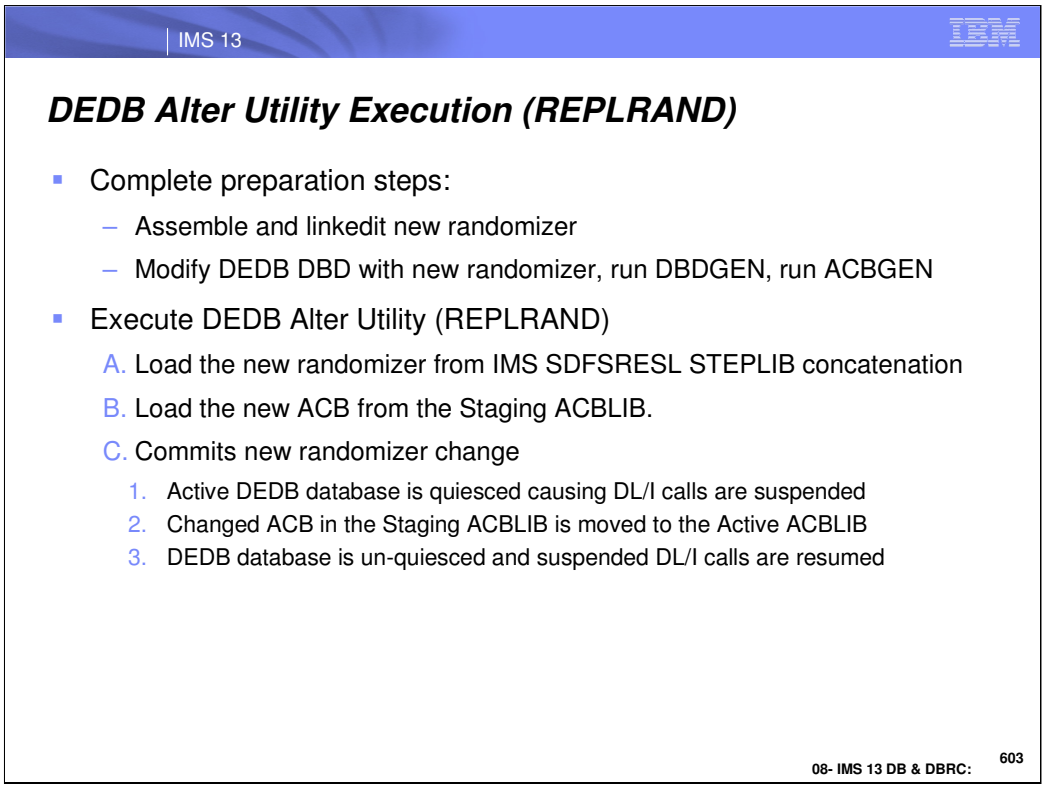
```
//ALTAREA  JOB ….
//FPUTIL   PROC  SOUT=A,RGN=1M,
//               DBD=,REST=00,DIRCA=002,
//               PRLD=,IMSID=,AGN=,SSM=,ALTID=
//FPU      EXEC  PGM=DFSRRC00,REGION=&RGN,
//               PARM=(IFP,&DBD,DBF#FPU0,&REST,00,,1,
//               &DIRCA,&PRLD,0,,,,&IMSID,&AGN,&SSM,,
//               &ALTID)
//STEPLIB  DD DSN=IMS.CRESLIB,DISP=SHR
//PROCLIB  DD DSN=IMSVS.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT,…
//S0       EXEC FPUTIL,RGN=1M,DBD=DEDBJN21,REST=00,IMSID=IMS1
//SYSIN    DD   *
TYPE       ALTER
REPLRAND
RETRY      NO
TIMEOUT    30
GO
/*
```


This is sample JCL showing the DEDB Alter execution control cards for the REPLRAND function.

DEDB Alter (REPLRAND Function)

- REPLRAND
  - Changes Randomizer in RMNAME DEDB DBD statement while DEDB online
    - New name must be 2-stage randomizer
    - Existing randomizer must be a 2-stage randomizer
    - New name must be different than original name active for DEDB area
    - After DEDB Alter, new randomizer replaces old randomizer
      - All Areas in DEDB database use new randomizer
  - Supports DEDB database with or without SDEPs

The DEDB Alter function, REPLRAND, allows the user to change the randomizer name using the RMNAME DEDB DBD while the DEDB remains online. The new name and the existing name must be a 2-stage randomizer and the new name must be different from the existing randomizer name. After the DEDB Alter completes, the new randomizer replaces the existing randomizer and all Active Areas in the DEDB database begin to use the new randomizer. There can be only one Active DEDB changing at a time. The REPLRAND function supports DEDB databases with and without SDEPs.

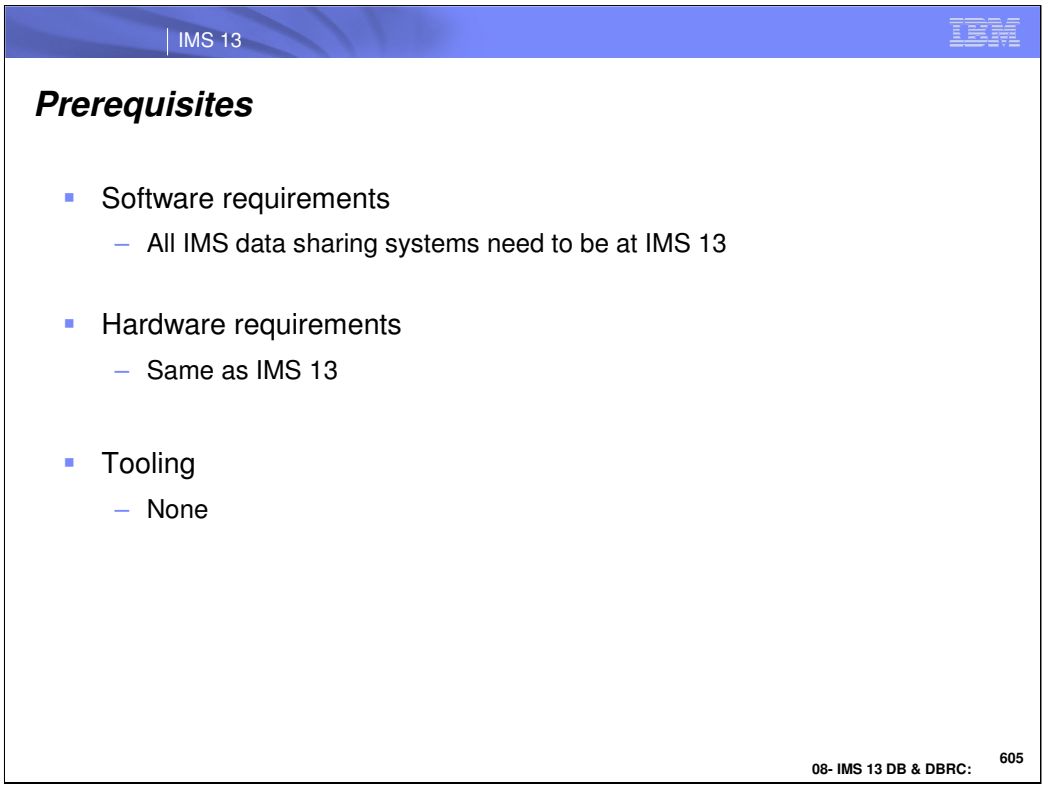


The DEDB Alter utility with the REPLRAND function executes after the new randomizer is assembled and linkedited, the DEDB DBD is modified with the new randomizer name, and the DBDGEN and ACBGEN are executed. The DEDB Alter REPLRAND function will load the new randomizer from the IMS SDFSRESL STEPLIB concatentation and the ACB from the Staging ACBLIB. When the randomizer change is committed, the Active DEDB database is quiesced forcing DL/I calls to be suspended and allowing the changed ACB in the Staging ACBLIB to be moved to the Active ACBLIB. Finally, the DEDB database is unquiesced and DL/I calls are resumed.

Post-DEDB Alter Utility Execution

- If DEDB Alter is Successful
  - New Randomizer replaces existing randomizer
- If DEDB Alter is Unsuccessful
  - Existing Randomizer remains in effect

If the DEDB Alter REPLRAND function is successful, the new randomizer replaces the existing randomizer. If the function is unsuccessful, the existing randomizer remains in effect.

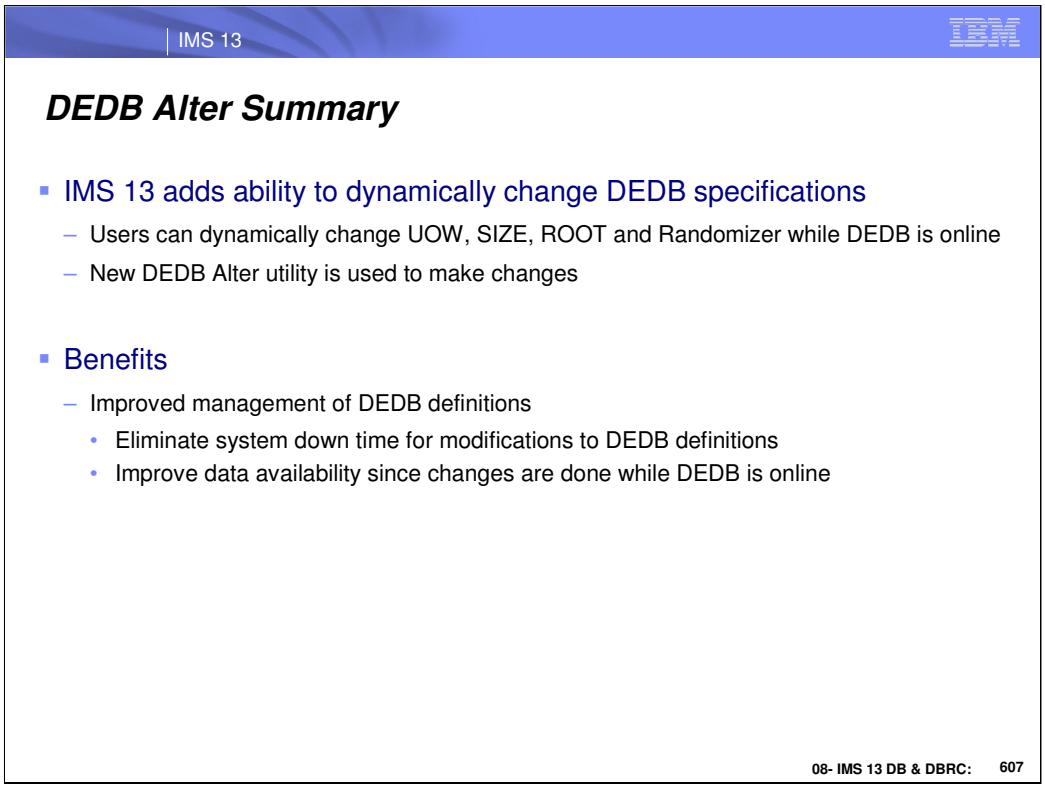


For the DEDB Alter functions, all IMS data sharing systems need to be at the IMS 13 level. There are no special hardware requirements for DEDB Alter.

Restrictions

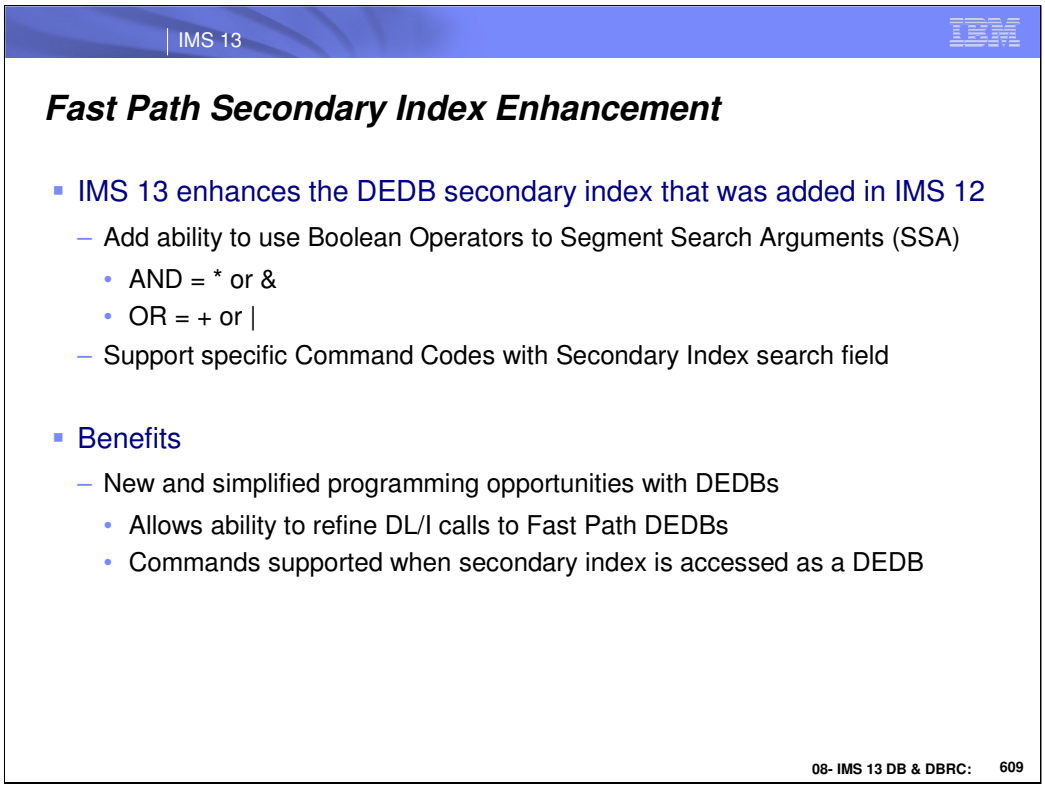
- Software restrictions
  - None
- Environment restrictions
  - The randomizer must be a 2-stage randomizer
    - RMNAME=(randomizer_name,2)
    - Logic must act like a 2-stage randomizer
  - DEDB Areas must be registered to DBRC
  - Supports ACBSHR=Y for sharing IMS system if it is an:
    - Active system for XRF
    - Active system for FDBR
  - MINVERS = 13.1

The randomizer must be a 2-stage randomizer where the logic in the randomizer behaves like a 2-stage randomizer. Also, the DEDB Area must be registered to DBRC. This supports ACBSHR=Y for the sharing IMS system if the active system is XRF or FDBR. Finally, the DBRC MINVERS parameter must be set to 13.1 to use the DEDB Alter capability.



In IMS 13, Fast Path has added the ability to dynamically change specific DEDB specifications. For example, the UOW, SIZE, ROOT, and Randomizer routine can be changed while the DEDB is online. There is a new DEDB Alter utility that allows these DEDB changes to occur. Allowing dynamic changes to the DEDBs will improve data availability and reduce system down time.

This section addresses the Secondary Index Enhancements new function.



In IMS 12, IMS Fast Path added the ability to create secondary indexes. In IMS 13, this function has been enhanced to allow Segment Search Arguments to use the Boolean Operators “AND” and “OR”. Also, support was added to allow specific Command Codes to be used with the Secondary Index search field.

This support allows better programming capabilities for DEDBs. It allows the ability to refine DL/I calls and additional command code support.

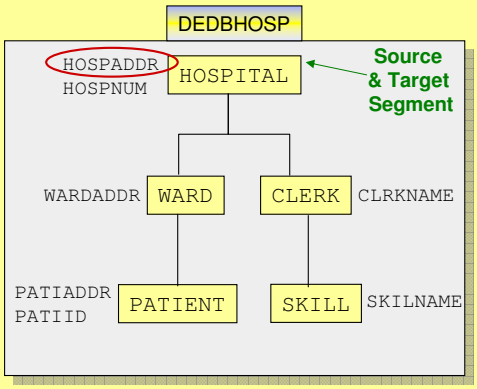
IMS 12 PTFs Forward Fitted to IMS 13

APAR	PTF	Description	Available in IMS 13
PM59166		Boolean Operator support for Fast Path Secondary Index DL/I calls	QPP Tape
PM49031		Command Code, Multiple SSA, Qualified Get Call support for FP Secondary Index DL/I calls (Target Segment = Root)	QPP Tape
PM59181		Command Code, Multiple SSA, Qualified Get Call support for FP Secondary Index DL/I calls (Target Segment ≠ Root)	Dec, 2012

There are three IMS 12 PTFs for Fast Path Secondary Index functions that are being forwarded into IMS 13.

Example: Target (Root) Segment = Source Segment

```
DBD1    DBD  NAME=DEDBHOSP,ACCESS=DEDB
AREA    DD1=GSAREA1…
SEGM    NAME=HOSPITAL,PARENT=0…
FIELD   NAME=(HOSPNAME,SEQ,U)…
FIELD   NAME=HOSPADDR…
FIELD   NAME=HOSPNUM…
LCHILD  NAME=(IXSASEG,FPSI1ASA),PTR=SYMB
XDFLD   NAME=IXSAIDX,SRCH=HOSPADDR
SEGM    NAME=WARD,PARENT=HOSPITAL…
FIELD   NAME=(WARDNAME,SEQ,U),…
FIELD   NAME=WARDADDR,…
SEGM    NAME=PATIENT,PARENT=WARD…
FIELD   NAME=(PATINAME,SEQ,U),…
FIELD   NAME=PATIADDR,…
FIELD   NAME=PATIID,…
SEGM    NAME=CLERK,PARENT=HOSPITAL…
FIELD   NAME=(CLERKNUM,SEQ,U)
FIELD   NAME=CLRKNAME,…
SEGM    NAME=SKILL,PARENT=CLERK…
FIELD   NAME=(SKILLNUM,SEQ,U)
FIELD   NAME=SKILNAME,…
DBDGEN
```



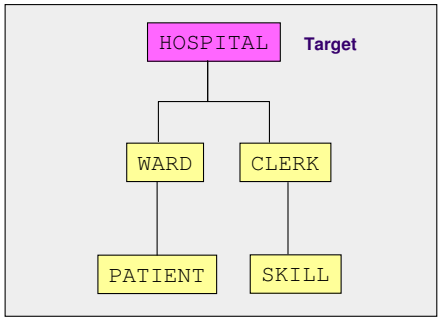
This is the DBD for the DEDB (DEDBHOSP). Segment HOSPITAL is the source and the target segment. The LCHILD and XDFLD statements follow the SEGM statement for HOSPITAL.

The LCHILD segment specifies the secondary index segment, IXSASEG, and database, FPSI1ASA. PTR=SYMB is specified, as required.

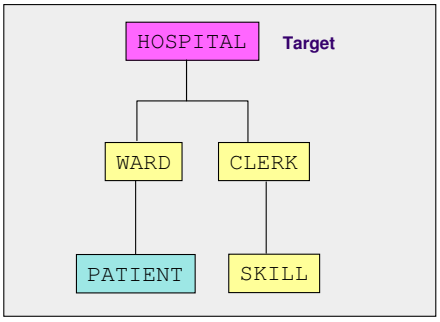
The XDFLD statement specifies the name of the search field, HOSPADDR, for use with the secondary index. It also specifies that field IXSAIDX is used to build the search field.

Indexing When Target Is Root Segment

Physical data structure



Secondary data structure



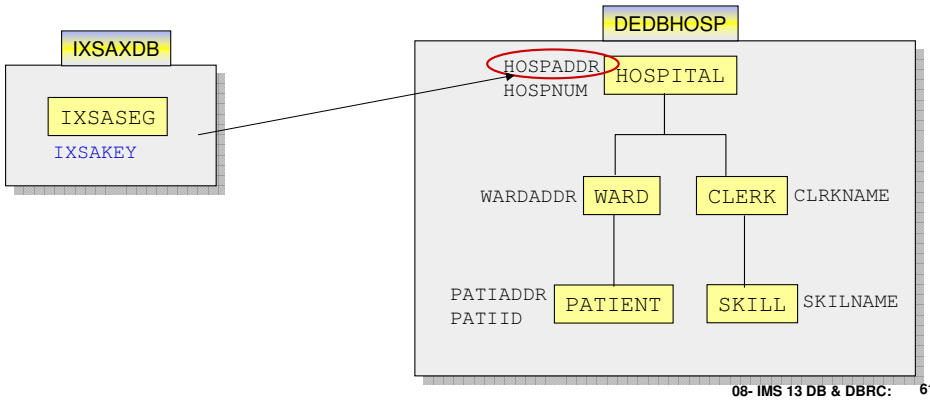

```
PCB TYPE=DB,…,PROCSEQD=…
SENSEG NAME=HOSPITAL,PARENT=0
SENSEG NAME=WARD,PARENT=HOSPITAL
SENSEG NAME=PATIENT,PARENT=WARD
SENSEG NAME=CLERK,PARENT=HOSPITAL
SENSEG NAME=SKILL,PARENT=SKILL
PSBGEN  PSBNAME=…,
END
```

Key Feedback for PATIENT:
SI key + key of WARD + key of PATIENT

This page illustrates the physical structure of a database and the structure as viewed when accessing the database through the secondary index. In this case, they are the same since the root segment is also the target segment. As explained on the previous page, the key feedback area is composed of the secondary index key and the keys of the dependent segments. The key feedback area for segment PATIENT is composed of the secondary index key, the key of segment WARD and the key of segment PATIENT.

Secondary Index DB (Target (Root) = Source Segment)

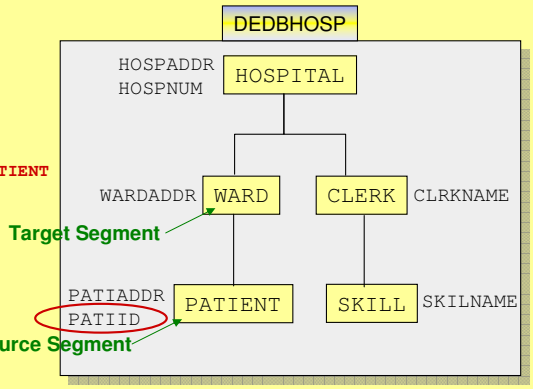
```
DBDSX   NAME=IXSAXDB,ACCESS=(INDEX,SHISAM)
DATASET DD1=FPSI1ASA,
SEGM    NAME=IXSASEG,PARENT=0,…
FIELD   NAME=(IXSAKEY,SEQ,U),…
LCHILD  NAME=(HOSPITAL,DEDBHOSP),INDEX=IXSAIDX,PTR=SYMB
DBDGEN
```



This is the DBD for the secondary index database. The SEGM statement defines the segment in the secondary index. The FIELD statement defines the sequence field in the secondary index. The LCHILD statement specifies the target segment, HOSPITAL, and database, DEDBHOSP, in the NAME= parameter. The INDEX= parameter specifies the NAME= value on the XDFLD statement of the target database. This is IXSAIDX, the search field for use with the secondary index.

Example: Target (Dependent) ≠ Source Segment

```
DBD1    DBD  NAME=DEDBHOSP,ACCESS=DEDB
AREA    DD1=GSAREA1…
SEGM    NAME=HOSPITAL,PARENT=0…
FIELD   NAME=(HOSPNAME,SEQ,U)…
FIELD   NAME=HOSPADDR…
FIELD   NAME=HOSPNUM…
SEGM    NAME=WARD,PARENT=HOSPITAL…
FIELD   NAME=(WARDNAME,SEQ,U),…
FIELD   NAME=WARDADDR,…
LCHILD  NAME=(IXSBSEG,FPSI1ASB),PTR=SYMB
XDFLD   NAME=IXSBIDX,SRCH=PATIID,SEGMENT=PATIENT
SEGM    NAME=PATIENT,PARENT=WARD…
FIELD   NAME=(PATINAME,SEQ,U),…
FIELD   NAME=PATIADDR,…
FIELD   NAME=PATIID,…
SEGM    NAME=CLERK,PARENT=HOSPITAL…
FIELD   NAME=(CLERKNUM,SEQ,U)
FIELD   NAME=CLRKNAME,…
SEGM    NAME=SKILL,PARENT=CLERK…
FIELD   NAME=(SKILLNUM,SEQ,U)
FIELD   NAME=SKILNAME,…
DBDGEN
```

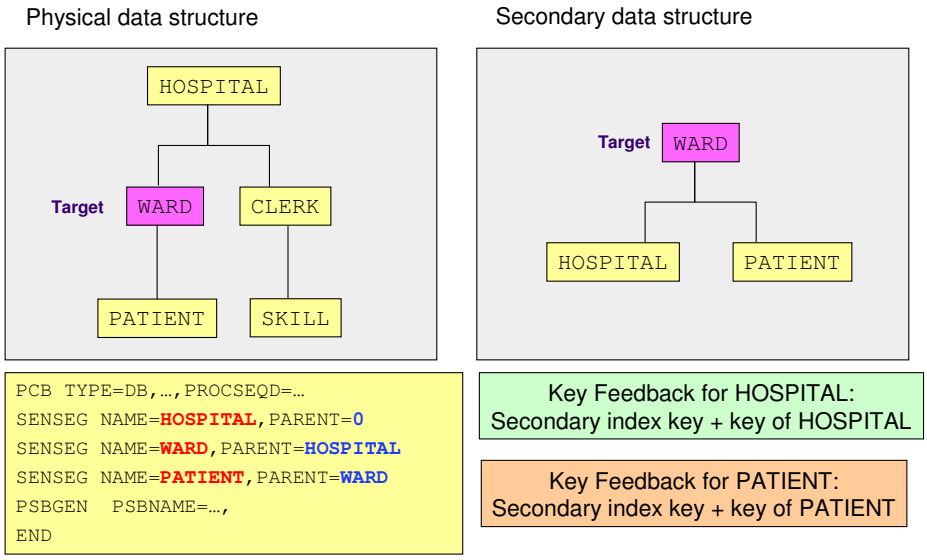


This is the DBD for the DEDB (DEDBHOSP). Segment HOSPITAL is the target and PATIENT is the source segment. The LCHILD and XDFLD statements follow the SEGM statement for HOSPITAL.

The LCHILD segment specifies the secondary index segment, IXSBSEG, and database, FPSI1ASB.

The XDFLD statement specifies the name of the search field, PATIID, for use with the secondary index. It also specifies that field IXSBIDX is used to build the search field.

Indexing When Target Is Dependent Segment




```
PCB  TYPE=DB,…,PROCSEQD=…
SENSEG  NAME=HOSPITAL,PARENT=0
SENSEG  NAME=WARD,PARENT=HOSPITAL
SENSEG  NAME=PATIENT,PARENT=WARD
PSBGEN   PSBNAME=…,
END
```

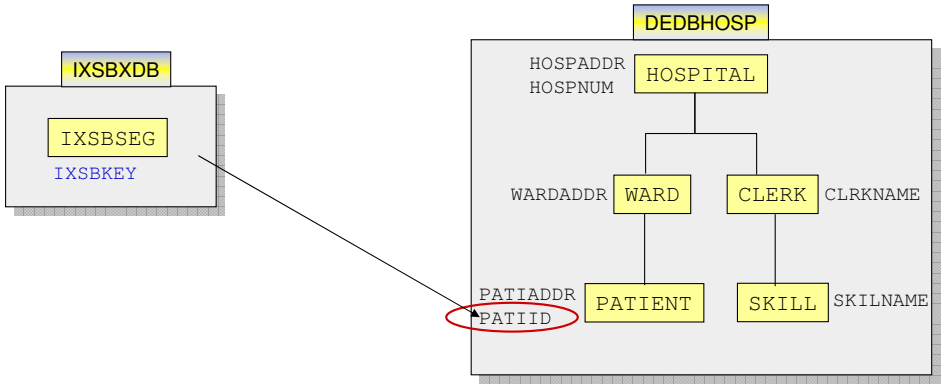
Key Feedback for HOSPITAL:
Secondary index key + key of HOSPITAL

Key Feedback for PATIENT:
Secondary index key + key of PATIENT

This page illustrates the physical structure of a database and the structure as viewed when accessing the database through the secondary index. In this case, they are not the same since the target segment is a dependent segment. The Key Feedback for HOSPITAL is the secondary index key plus the key of HOSPITAL. The Key Feedback area for PATIENT is the secondary index key plus the key of PATIENT.

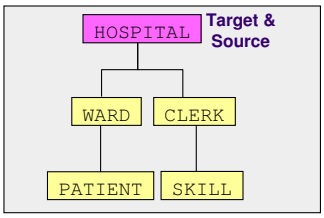
Secondary Index DB (Target (Dependent) ≠ Source Segment)

```
DBDSX   NAME=IXSBXDB,ACCESS=(INDEX,SHISAM)
DATASET DD1=FPSI1ASB,
SEGM    NAME=IXSBSEG,PARENT=0,…
FIELD   NAME=(IXSBKEY,SEQ,U),…
LCHILD  NAME=(PATIENT,DEDBHOSP),INDEX=IXSBIDX,PTR=SYMB
DBDGEN
```



This is the DBD for the secondary index database. The SEGM statement defines the segment in the secondary index. The FIELD statement defines the sequence field in the secondary index. The LCHILD statement specifies the target segment, PATIENT, and database, DEDBHOSP, in the NAME= parameter. The INDEX= parameter specifies the NAME= value on the XDFLD statement of the target database. This is IXSBIDX, the search field for use with the secondary index.

Boolean Support (AND= or &, OR=+ or |)*

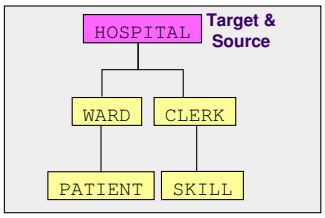


Example Statement	Result
GU HOSPITAL (IXSAIDX =Elm Street| IXSAIDX =Doe Street)	Get the Hospital information for the hospital on “Elm Street” or “Doe Street”.

Note:
✓Boolean operators work with FPSI accessed as a database (ACCESS=INDEX).

The secondary index search field is now allowed to use Boolean Operators. These examples show how the Boolean Operators (* or &) and (+ or |) are used when the target and source segment are the root segment and when the source segment is a dependent segment.

Multiple SSA Support for Secondary Index



Example Statement	Result
GU HOSPITAL (IXSAIDX =Elm Street) WARD (WARDNAME=Emergency)	Get the Hospital information on "Elm Street" and "Emergency Ward" information

Restrictions: Does not support "Qualify on Any Field Name" support.

With Multiple SSA support for a secondary index, it is possible to use the index search field along with other SSA arguments.

Supported Command Codes

Command Code	Explanation
C	Use the concatenated key of a segment to identify the segment
D	Retrieve or insert a sequence of segments in a hierarchic path using only one call, instead of using a separate (path) call for each segment.
F	Back up to the first occurrence of a segment under its parent when searching for a particular segment occurrence. Disregarded for a root segment.
L	Retrieve the last occurrence of a segment under its parent.
N	Designate segments that you do not want replaced when replacing segments after a Get Hold call.
P	Set parentage at a higher level than what it usually is (the lowest-level SSA of the call).
Q	Reserve a segment so that other programs cannot update it until you have finished processing and updating it.
U	Limit the search for a segment to the dependents of the segment occurrence on which position is established.
V	Use the hierarchic level at the current position and higher as qualification for the segment.
-	Null. Use an SSA in command code format without specifying the command code.

In IMS 13, Command Code support was added to the Fast Path DEDB Secondary Index capability. The codes that are supported are: C, D, F, L, N, P, Q, U, V, and – “null”.

Unsupported Command Codes

Command Code	Explanation	Status Code
A	Clear positioning and start the call at the beginning of the database	AJ
G	Prevent randomization and search the database sequentially	AJ
M	Move a subset pointer to the next segment occurrence after your current position	AJ
O	Qualify by Position	AD
R	Retrieve the first segment occurrence in a subset	AJ
S	Unconditionally set a subset pointer to the current position	AJ
W	Set a subset pointer to your current position, if the subset pointer is not already set	AJ
Z	Set a subset pointer to 0, so it can be reused	AJ

In IMS 13, the codes that are not supported are: A, G, M, O, R, S, W, and Z.

Fast Path Secondary Index Enhancement

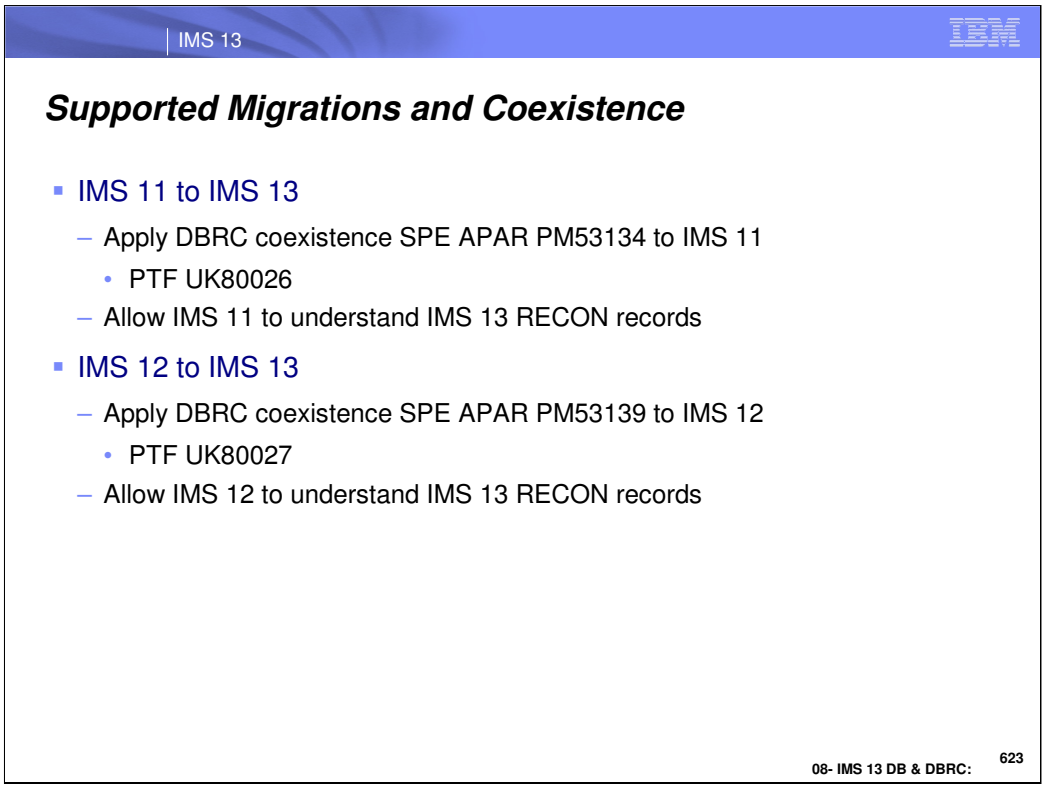
- IMS 13 enhances the DEDB secondary index that was added in IMS 12
  - Add ability to use Boolean Operators to Segment Search Arguments (SSA)
    - AND = * or &
    - OR = + or |
  - Support specific Command Codes with Secondary Index search field
    - Supported: C, D, F, L, N, P, Q, U, V, - (NULL)
    - Unsupported: A, G,M, R, S, W, Z
  - Supports User Data Partitioning environments
- Benefits
  - New and simplified programming opportunities with DEDBs
    - Allows ability to refine DL/I calls to Fast Path DEDBs
    - Commands supported when secondary index is accessed as a DEDB

In IMS 12, IMS Fast Path added the ability to create secondary indexes. In IMS 13, this function has been enhanced to allow Segment Search Arguments to use the Boolean Operators “AND” and “OR”. Also, support was added to allow specific Command Codes to be used with the Secondary Index search field.

This support allows better programming capabilities for DEDBs. It also allows the ability to refine DL/I calls and use command code support.

DBRC Migration and Coexistence

This sections covers the DBRC Migration and Coexistence function in IMS 13.



IMS 11 RECONs may be upgraded directly to IMS 13. Similarly, IMS 12 RECONs may be upgraded to IMS 13. There is no support to upgrade RECONs from previous releases directly to IMS 13.

PM53134 is an IMS 11 SPE (Small Programming Enhancement) APAR. It allows IMS 11 to use RECONs which have been upgraded to IMS 13. The PTF associated with this SPE is UK80026.

PM53139 is an IMS 12 SPE APAR. It allows IMS 12 to use RECONs which have been upgraded to IMS 13. The PTF associated with this SPE is UK80027.

These APARs should be applied to IMS 11 or IMS 12 before its RECONs are upgraded to IMS 13.

RECON Listings

- "COEXISTENCE LEVEL" in subsystem record listing
  - Added by IMS 10
  - May be used to determine if subsystems would cause an upgrade failure

```
SSYS
 SSID=IMS1       LOG START=12.267 12:45:47.2
 SSTYPE=ONLINE   ABNORMAL TERM=OFF   RECOVERY STARTED=NO    BACKUP=N
 TRACKED=NO      TRACKER TERM=OFF    SHARING COVERED DBS=NO
 IRLMID=**NULL**   IRLM STATUS=NORMAL         GSNAME=**NULL**
 COEXISTENCE LEVEL=13.1

 AUTHORIZED DATA BASES/AREAS=4      VERSION=11.1 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 11.1 but has the 13.1 coexistence maintenance applied

IMS 10 added the coexistence level to the RECON listing of subsystem records. The VERSION= field 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 13 DBRC coexistence maintenance has been applied to the IMS 11 system used by this subsystem. This listing could have been produced by an IMS 11 or IMS 12 DBRC utility with the IMS 13 coexistence SPE applied or it could have been produced by the IMS 13 DBRC utility.

CHANGE.RECON UPGRADE (IMS V11 to V13 Only)

- Upgrade reads all database records to ensure that the high-order bit is on in all DMB numbers
  - If high-order bit is not on (should not occur)
    - High order bit is turned on if database is not authorized
      - Message issued:

        ```
        DSP1235W THE INTERNAL REPRESENTATION OF THE DMB NUMBER
        FOR DATABASE xxxxxxxx IS INCORRECT
        ```
    - High order bit is not turned on if database is authorized
      - Message issued:

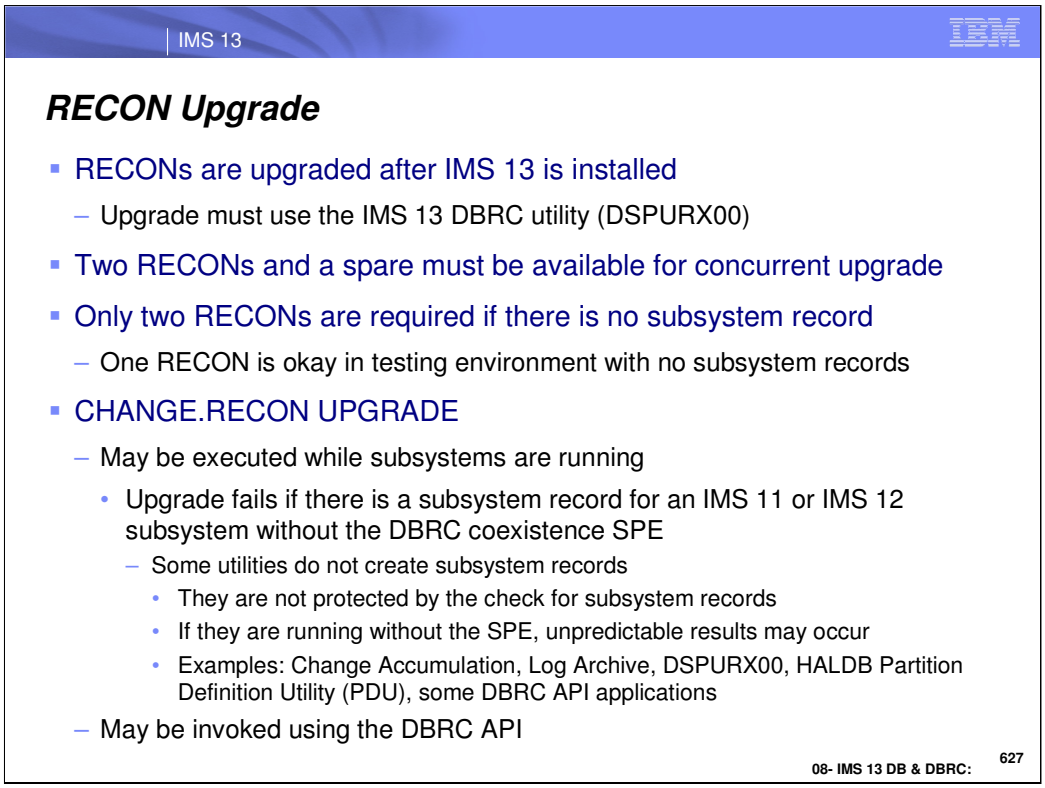
        ```
        DSP1236E THE INTERNAL REPRESENTATION OF THE DMB NUMBER
        FOR DATABASE xxxxxxxx COULD NOT BE CORRECTED BECAUSE
        THE DATABASE IS AUTHORIZED
        ```
        - This condition causes upgrade to fail

IMS 13 RECON upgrade uses the same check on the DMB table that was introduced for IMS V12. This check is needed only when the IMS V11 RECON is upgraded directly to IMS V13. It verifies that the high-order bit is on for all DMB numbers in database records. By convention, this bit should always be on. If it is not on, the upgrade process turns the bit on if the database is not authorized. If it is authorized, the upgrade cannot turn the bit on. When a DMB number is found without the high-order bit on, the upgrade issues either the DSP1235W or DSP1236E message. DSP1235W is issued when the upgrade is able to correct the bit setting. DSP1236E is issued when the upgrade cannot correct the bit setting because the database is authorized. If DSP1236E is issued, the upgrade must be done when the database is not authorized.

CHANGE.RECON UPGRADE CHECKUP

- CHECKUP keyword verifies RECON Upgrade will work
  - New in IMS 12
  - Upgrade is not done
  - Maybe used to check if upgrade would be successful
    - Only reads records which could stop upgrade
  - Messages are issued indicating whether upgrade would be successful
    - `DSP1238I RECON UPGRADE CHECKUP IS BEGINNING`
    - `DSP1239I RECON UPGRADE CHECKUP COMPLETED WITH NO ERRORS FOUND`
      - RC=0
      - RC=4
        - When DSP1235W issued (found high-order bit off in the DMB number and the database or area is not authorized)
    - `DSP1240E RECON UPGRADE CHECKUP COMPLETED AND FOUND ERROR RC=12`
      - DSP1236E issued (found high-order bit off in the DMB number and the database or area is authorized)

IMS 12 added the CHECKUP keyword for the CHANGE.RECON UPGRADE command. When CHECKUP is included in the command, an upgrade is not done; however, all records which could prevent an upgrade from being successful are read. This includes the database records mentioned on the previous page. The DFS1235W and DFS1236E messages are issued when the high-order bit of database records are not on.

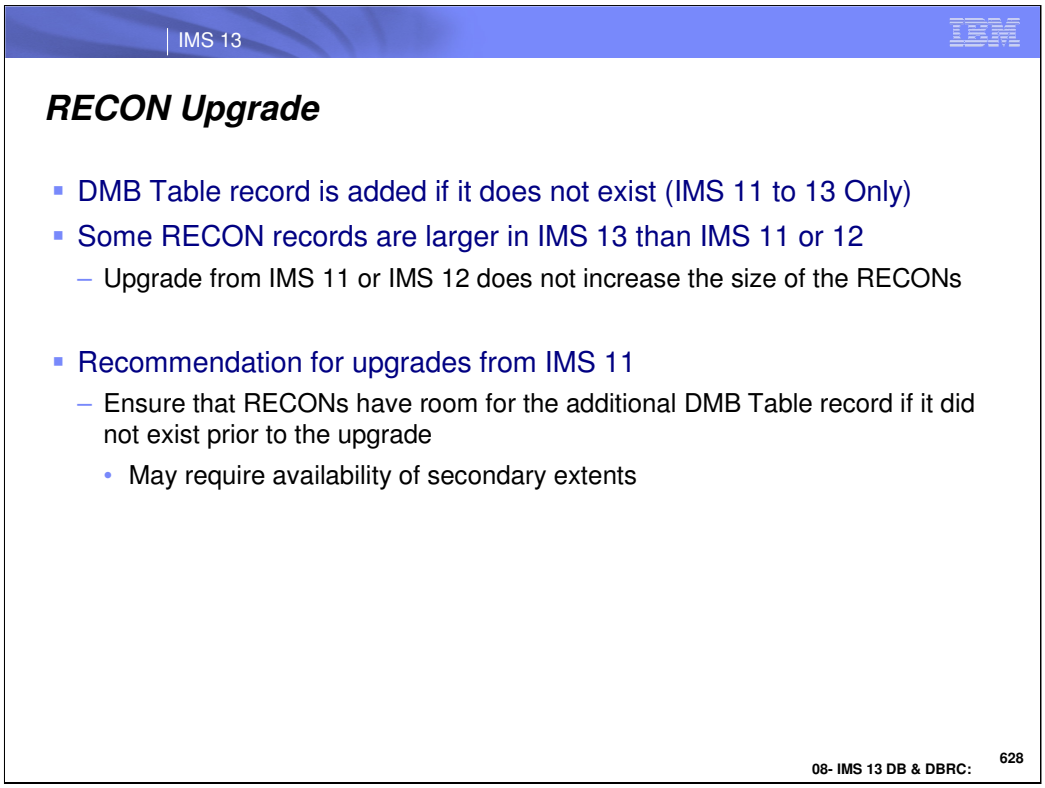


RECONs are upgraded to IMS 13 by using the DBRC CHANGE.RECON UPGRADE command with the IMS 13 DBRC utility (DSPURX00).

The concurrent upgrade process requires that there are two active RECON data sets with an available spare. On the other hand, if there are no subsystem records, the upgrade may be done without a spare RECON. The upgrade process upgrades the records in COPY1 and then makes COPY2 equal to COPY1. In a testing environment, you could do the upgrade with only one RECON (STARTNEW=YES), but this is not recommended for production environments.

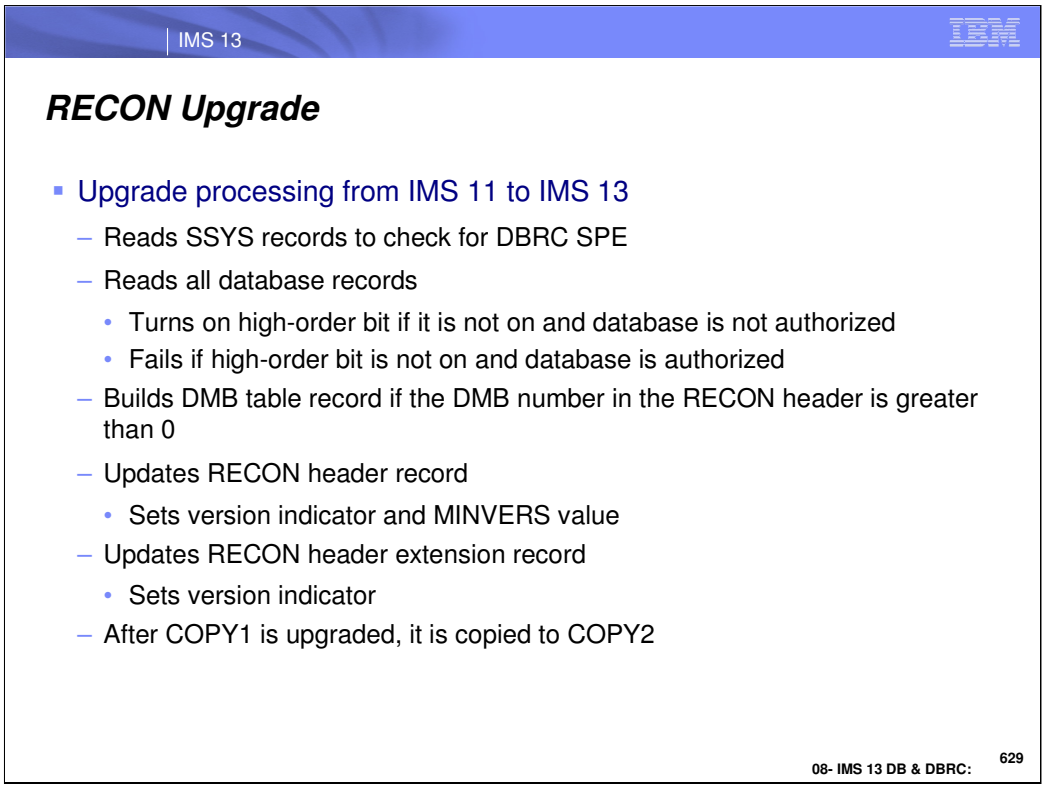
The upgrade may be run while the RECONs are allocated to and being used by IMS 11 or IMS 12. Of course, these systems must be able to use IMS 13 RECONs. The upgrade checks the RECONs to ensure that any subsystems using the RECONs are capable of using IMS 13 RECONs. It does this by examining the SUBSYS 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 (PM53134 or PM53139) which allows it to read IMS 13 RECONs.

IMS 10 added the capability to issue DBRC commands from programs using the DBRC API. This includes the capability to issue the CHANGE.RECON UPGRADE command.



The upgrade will add a DMB Table record if it does not already exist. It may not exist if the upgrade is from IMS V11 to IMS V13, however, it will already exist if the upgrade is from IMS V12 to IMS V13.

The upgrade of RECONs from IMS V11 or IMS V12 to IMS V13 will not increase the size of some RECON records. The RECON Partition record is only extended if HALDB alter is active in V13 with MINVERS set to 13.1.



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.

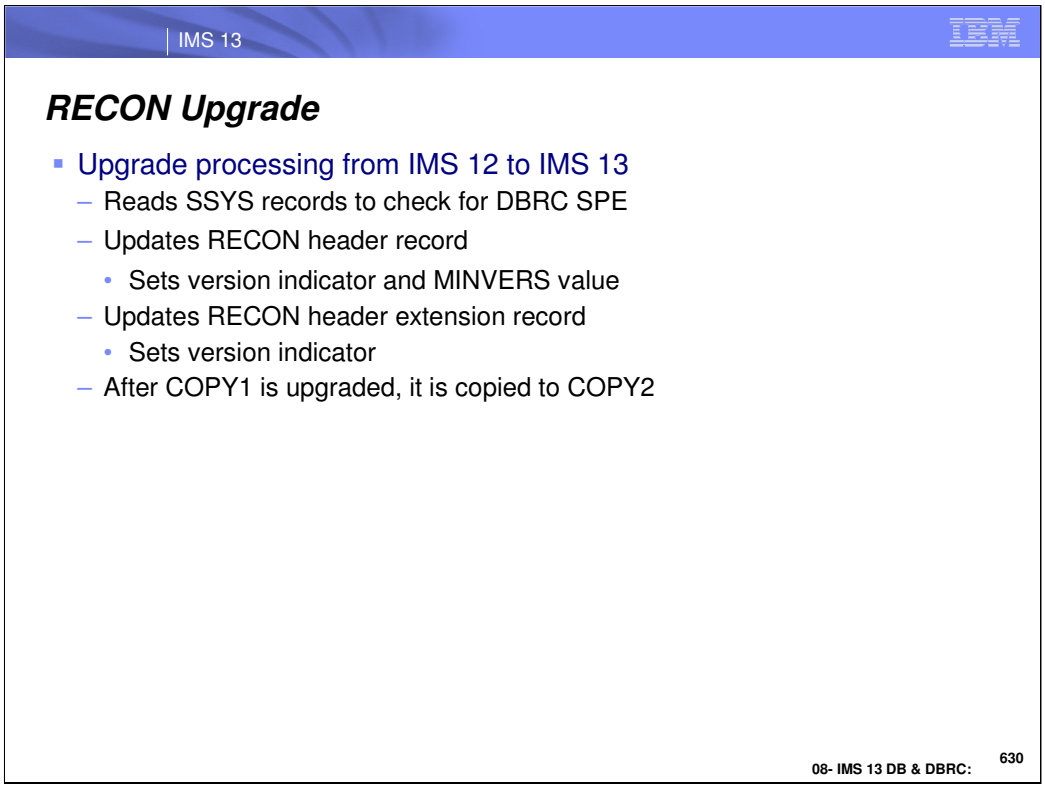
The update changes a few records in the RECONs.

All database records are read. A check is made to ensure that the high-order bit of the DMB numbers is on. If it is not on and the database is not authorized, the bit is turned on. If the high-order bit is not on and the database is authorized, the upgrade fails. The count of database records is kept. If the DMB table does not exist, it is built. If it does exist, it is rebuilt. The count of database records is kept.

The version indicator is set to 13 and the MINVERS value is set to ‘11.1' if it previously was ‘10.1'. The Cross DBRC Service Level ID (CDSLID) is set to the higher of the value in the RECONs before the upgrade and "1".

The version indicator in the RECON header extension record is set to 13.

The upgrade is done by upgrading the records in COPY1 and then copying it to COPY2.



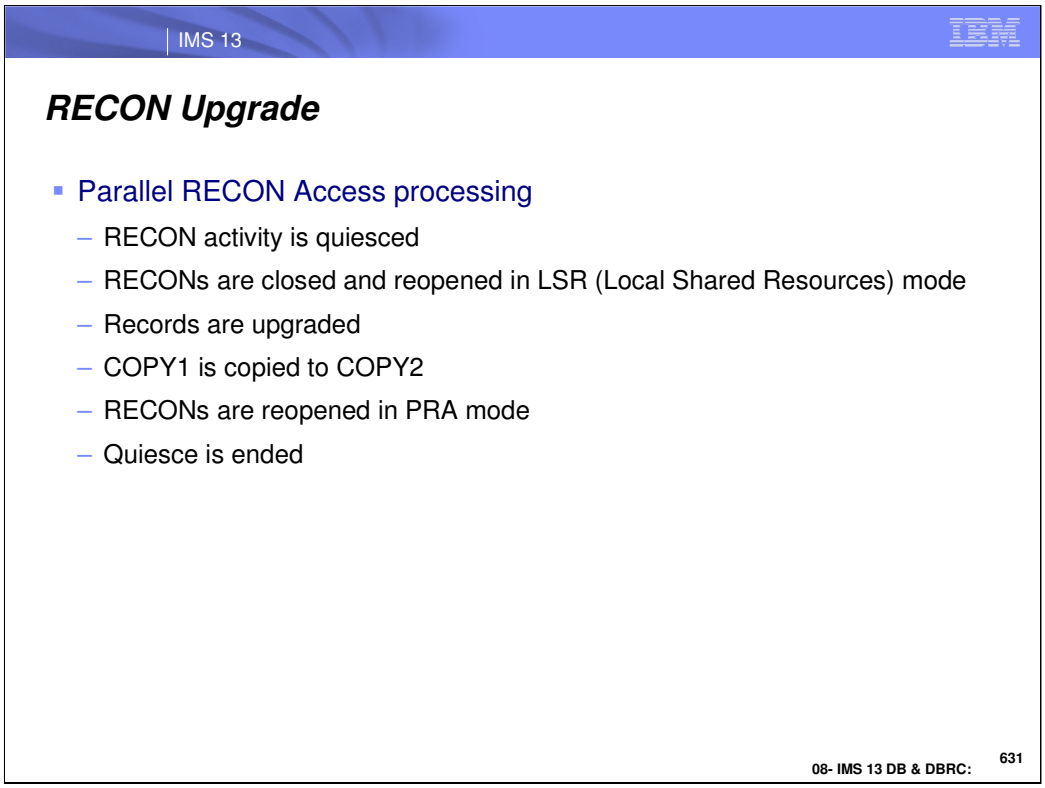
The upgrade of the RECONs includes the reading of the subsystem (SSYS) records to ensure that these subsystems are running with the DBRC coexistence SPE. If not, the subsystem could not use the RECONs and the upgrade fails.

The update changes the RECON header and RECON header extension records in the RECONs.

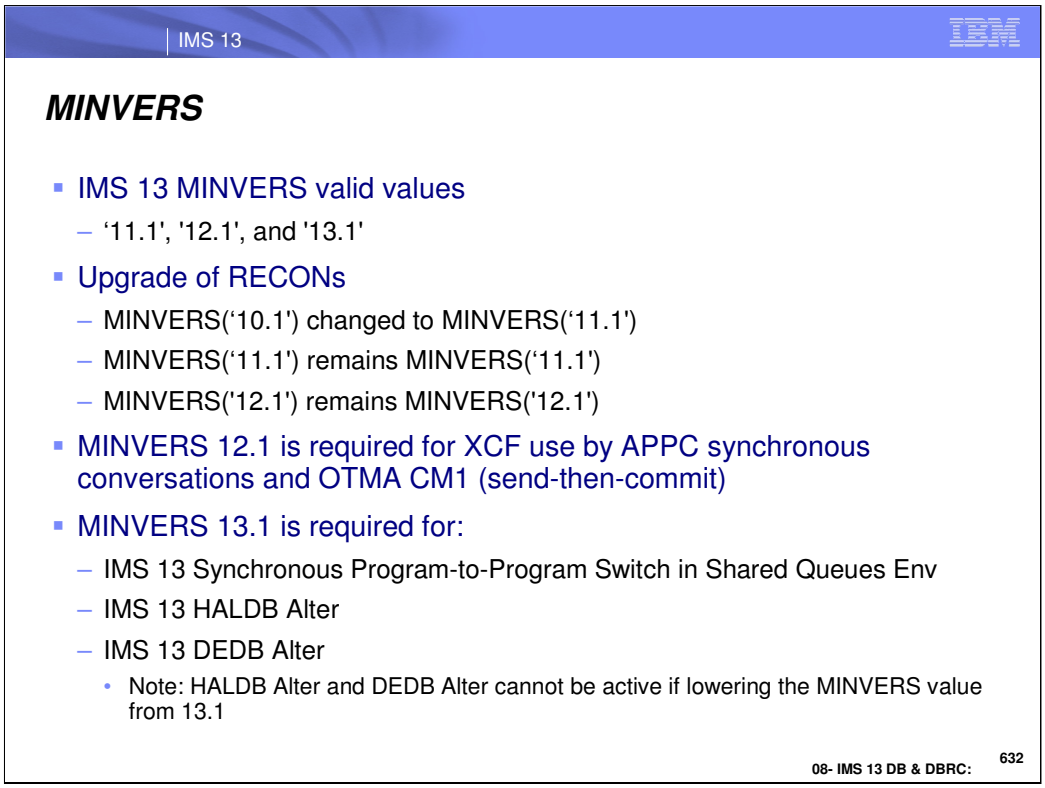
The header record is changed to set the MINVERS value to ‘11.1' if it previously was below ‘11.1'.

The version indicator in the RECON header extension record is set to 13.

The upgrade is done by upgrading the records in COPY1 and then copying it to COPY2.



If Parallel RECON Access is in effect, there cannot be any shunted I/O when the upgrade begins. The process begins with a quiesce close and a check for shunted I/O. The RECONs are closed and reopened in LSR mode. The records are upgraded as they are for non-PRA. This includes upgrading the records in COPY1 and then copying COPY1 to the spare. After the upgrade completes, the RECONs are reopened in PRA mode and the quiesce is ended.



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 IMS 13 RECONs is IMS 11. If the previous MINVERS value was for '10.1', it is changed to '11.1' by the upgrade. Otherwise, upgrades do not change the MINVERS value.

When the RECONs are upgraded to IMS 13 the minimum MINVERS value is 11.1.

MINVERS 12.1 is required for XCF use (instead of RRS) by APPC synchronous conversations and OTMA CM1 (send-then-commit).

MINVERS 13.1 is required for the new HALDB Alter, DEDB Alter, and Synchronous Program-to-Program Switch in a Shared Queues environment support added in IMS V13.

HALDB alter and DEDB alter cannot be active when lowering the MINVERS value from 13.1. New error messages are issued and the command fails.

DSP1249E MINVERS VALUE IS INCONSISTENT WITH THE HALDB ALTER STATUS FOR DATABASE name ALTER COUNT=xxxxx ALTER COMPLETE COUNT=xxxxx

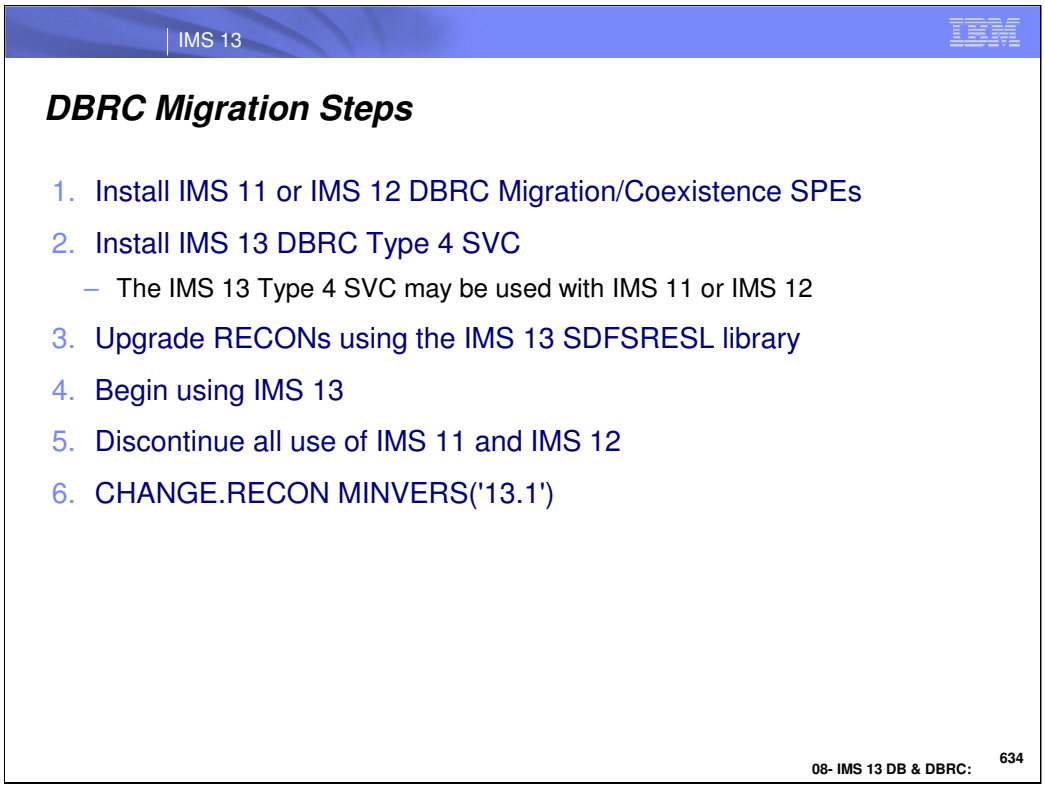
DSP1250E MINVERS VALUE IS INCONSISTENT WITH THE HALDB ALTER STATUS FOR PARTITION DATABASE name

DSP1251E MINVERS VALUE IS INCONSISTENT WITH THE DEDB ALTER STATUS FOR DEDB name ALTER COUNT=xxxxx

Log Archive (DFSUARC0) Region Size

- After RECONs are upgraded to IMS 13, then IMS 12 or IMS 11 Log Archive jobs will use additional memory
  - Both the IMS 13 and either IMS 11 or IMS 12 versions of RECON records are kept in memory
    - DBRC converts the records to IMS 11 or IMS 12 for processing by IMS 11 or IMS 12
- Recommendation
  - Use REGION=0M for IMS 11 and IMS 12 archive jobs

When the RECONs are at a higher level than the Log Archive utility, the utility keeps two copies of each RECON record. One is at the higher level. It is read from the RECONs or written to the RECONs. The other copy is at the lower level. It is processed by the utility. Records are converted from one level to the other when necessary. The second copy of each RECON record uses extra memory. This increases the memory requirement for the utility when it uses RECONs at a higher level.

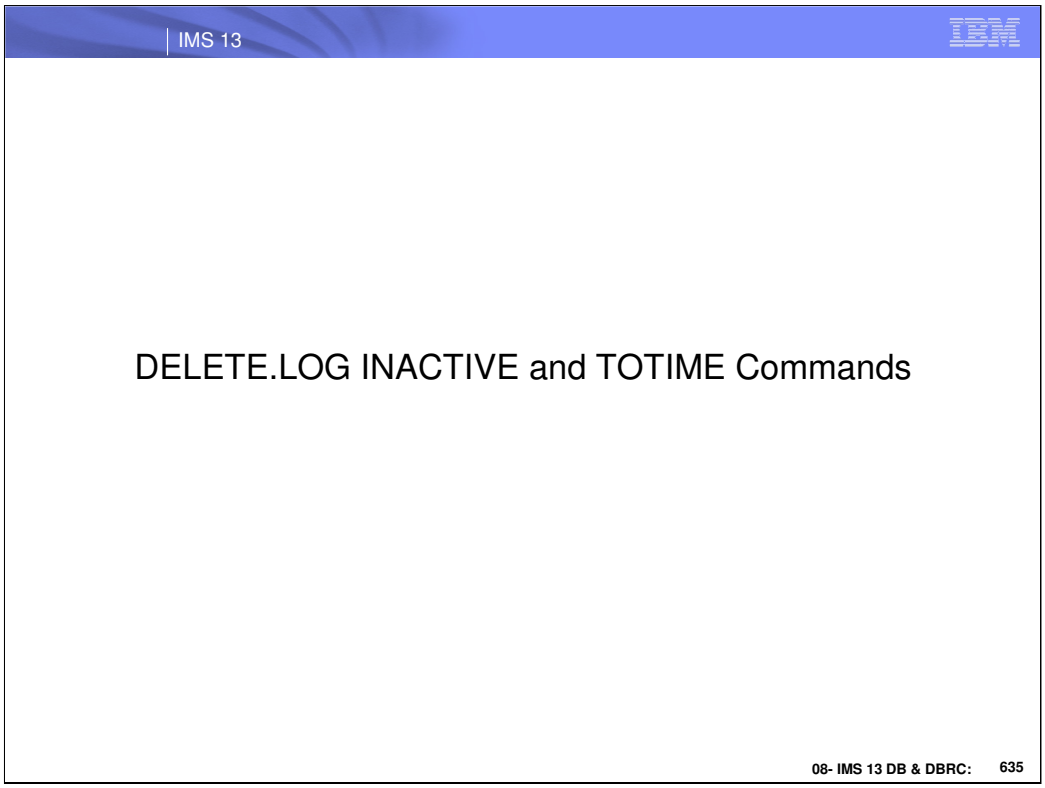


This shows the DBRC steps for migration to IMS 13.

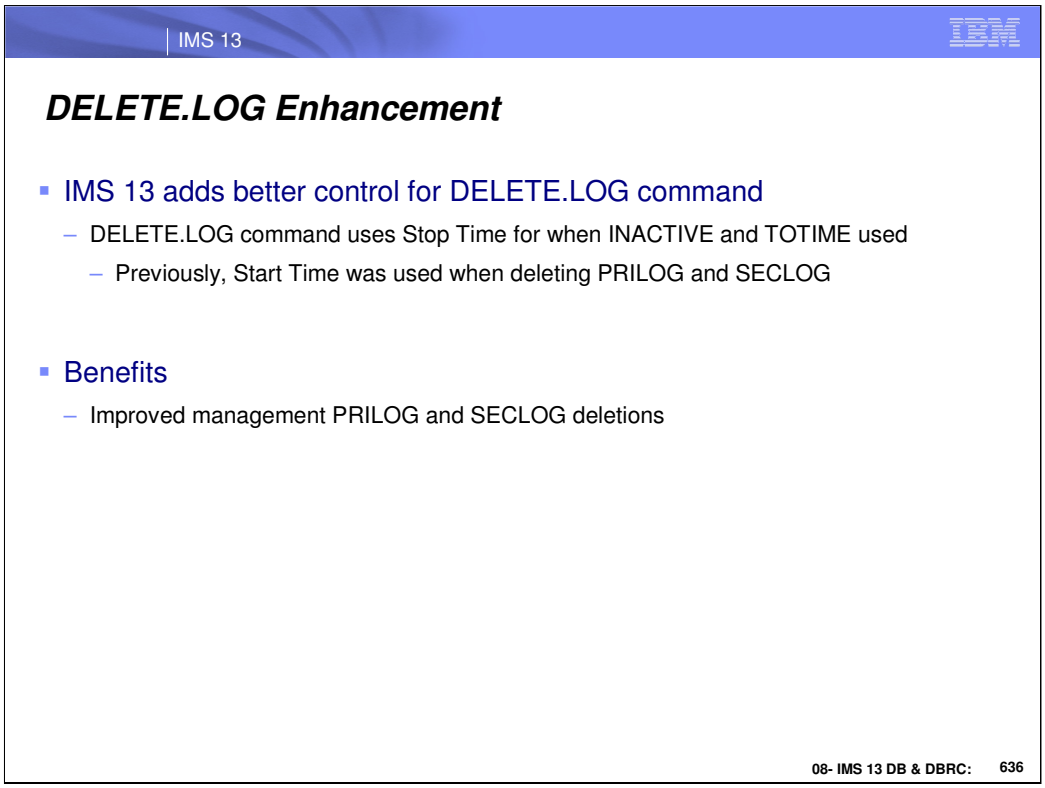
The first set of steps allows you to begin using IMS 13. The migration/coexistence SPE must be installed on the old release before you upgrade the RECONs to IMS 13. The IMS 13 DBRC Type 4 SVC must be installed before you may use IMS 13. The upgrade of the RECONs to IMS 13 requires that you use the SDFSRESL library created by the installation of IMS 13. The upgrade using this library will be to the IMS 13 format. Once the RECONs have been upgraded, you may begin using IMS 13. You may also continue to use IMS 11 or IMS 12.

When you upgrade the RECONs to IMS 13, the MINVERS value will be ‘11.1’ or higher.

Once you have discontinued all use of IMS 11 and IMS 12, you can change the MINVERS value to '13.1'.



This section addresses the DELETE.LOG function in IMS 13.



In IMS 13, the DELETE.LOG INACTIVE or DELETE.LOG TOTIME commands used the Start Time to determine when to delete the PRILOG and SECLOG records.

DELETE.LOG

- Problem description
  - DELETE.LOG command did not honor LOGRET setting
  - DELETE.LOG determines the timestamp for deleting logs
    - When PRILOG or SECLOG are closed
      - Start Time was used to determine when to delete PRILOG or SECLOG
      - PROBLEM:
        - Stop Time may still be in LOGRET period
- IMS 13 Solution:
  - DELETE.LOG INACTIVE and DELETE.LOG TOTIME
    - Will check Stop Time to determine if closed PRILOG or SECLOG is deleted
      - Must be no updates on any databases on the PRILOG or SECLOG

The DELETE.LOG command determines when the PRILOG or SECLOG records are deleted. There must be no updates on any databases for the logs in question. Prior to IMS 13, the DELETE.LOG command always used the log Start Time. In IMS 13, DBRC will check to see if the Stop Time is beyond the LOGRET time to determine if the PRILOG or SECLOG should be deleted.

(this page intentionally left blank)