IMS Catalog

Information Management software

Agenda

- IMS Catalog Overview
- IMS Catalog Enablement
- IMS Catalog Lifecycle
- IMS Catalog and ACBLIB Migrations
- Supplement - IMS Catalog Sharing

IMS Catalog Overview

Background

- Customer's need for IMS Metadata not being met
  - No trusted online source for IMS database, program and application metadata information
- Customers can't leverage newer IMS database functions
  - Large scale deployment of IMS Open Database is a challenge
  - JDBC metadata discovery APIs are insufficient
    - can't visualize and report on databases, generate SQL queries, query database contents, generate pureQuery applications, etc…
- Customers limited to database metadata generated by DLIModel Utility
  - Current Java metadata classes are …
    - offline and not guaranteed to be current
    - deployed everywhere
    - hard to maintain and keep in sync with realtime IMS resources
    - not trusted and inflexible
    - not easy to manage or scale for large Open Database solutions
    - lacking application information

Today, there is no trusted online source for both IMS Database and Application metadata. IMS database metadata is loosely scattered across application source and IMS resource definitions. The ACBLIB contains some trusted information about the IMS databases, but it's not enough. The metadata isn't complete and is in an IMS proprietary format. This limits the ability of IMS to offer metadata discovery and exchange. This results in impacts in the areas of integration, impact analysis, and Open Database management.

Large scale deployment of Open Database solution is possible but requires a significant amount of work on the customer's part. Customers need to generated the local metadata files with the DLIModel utility and distribute a copy of the files to where ever the application(s) lives. If the customer is running an application on a large number of servers, then each server will need a local copy of the metadata. Changes made to the database or application metadata will need to be regenerated in the DLIModel utility and then propagated to each server.

The metadata information generated from the DLIModel Utility is offline and not guaranteed to be current.

Many tools in the IBM portfolio leverage the catalog of a DBMS for impact analysis, modeling and metadata discovery. The lack of a catalog limits the ability of IMS to integrate with these solutions.

It is difficult for IMS customers to calculate the impact of a change to the physical structure of a database. The catalog will capture, among other things, the relationships between PSBs and DBDs. The catalog will be able to provide information such as the following:

Which DBDs are referenced by the PSB named 'XYZ'?

Which database segments use exit 'ABC'?

What is the data type of field 'FLDA' in segment 'SEG1' of DBD 'DB1'?

Due to the lack of an IMS catalog, Open Database users are required to generate a Java metadata class using the DLIModel utility that represents the metadata for a given Program Specification Block (PSB) and related Database Definitions (DBDs). The Java class must be maintained and kept in sync with the target IMS resources being accessed. This results in added management and complexity.

Solution

- Provide a trusted, online source for IMS database, program and application metadata information →
  - IMS catalog
  - Store the IMS catalog in an IMS database
- Allow for better scalability of Open Database solutions by removing the local metadata requirement

The IMS Catalog:

- Provides a trusted, online source for IMS database and application metadata information which is fully managed by IMS
- Uses an IMS Database for storing the metadata information
- Allows for better scalability of Open Database by removing the local metadata requirement

The IMS catalog is stored in an IMS database and is accessible via both standard JDBC/SQL and traditional DLI access.

The Universal JDBC drivers will leverage the catalog for metadata exchange and discovery, enabling a more flexible and scalable Open Database solution.

The IMS Explorer will also use the catalog for metadata exchange and discovery. IMS Explorer offers physical modeling of IMS PSBs and DBDs; it will be able to connect to the IMS catalog in order to get a trusted and comprehensive view of the database and application information.

The IMS catalog represents a significant component of the IMS simplification and integration strategy. It will offer the foundation for possible future enhancements such as dynamic database change and database versioning.

Benefits

- Business Value
  - Offers a trusted and comprehensive view of IMS database metadata managed by IMS using standard interfaces
    - IMS Universal Drivers (SQL and DLI)
    - Traditional IMS DLI database queries
  - Opens up metadata discovery and exchange for IMS Open Database and the IMS Explorer
  - Permits future IMS integration with IBM tools
    - QMF
    - COGNOS
    - Optim Development Studio
    - Rational Asset Analyzer
    - InfoSphere Data Architect
  - Enables scalable and flexible IMS Open Database solutions
    - Applications no longer need to maintain local Java metadata
    - Applications can reference the IMS catalog

The IMS catalog offers a trusted and comprehensive view of IMS database metadata and is fully managed by IMS. It is an IMS database which is accessible via standard JDBC/SQL and traditional DLI interfaces.

The IMS Universal JDBC Drivers can be used to access the catalog to query IMS metadata. The Universal Drivers will leverage the catalog for metadata exchange and discovery, enabling a more flexible and scalable Open Database solution. IMS Open Database users will no longer need to maintain the Java metadata class but can instead reference the IMS catalog.

The IMS Explorer can use the catalog for metadata exchange and discovery. IMS Explorer offers physical modeling of IMS PSBs and DBDs; it can connect to the IMS catalog in order to get a trusted and comprehensive view of the IMS database and application information. IMS Explorer can also be used to import application metadata via DBD source macros which can then be incorporated into the catalog.

With the catalog, IMS can be integrated with various IBM solutions. Business solutions which require the discovery or exchange of metadata for impact analysis or data modeling can potentially exploit the IMS catalog.

COGNOS, Rational Asset Analyzer, Optim Development Studio, Data Source Explorer, and Data Project Explorer can leverage the JDBC metadata discovery APIs to visualize and report on the target database, generate SQL queries, query database contents, generate pureQuery applications, etc. With the catalog, IMS can be just another supported data source with the same look and feel of other supported data sources.

What is Metadata?

B
- Business metadata
  - Business rules, definitions, terminology, glossaries, algorithms and lineage using business language
  - Audience: Business users

T
- Technical metadata
  - Defines source and target systems
    - Table & Column / Segment & Field structures and attributes
    - Derivations and dependencies
  - Audience: Specific tool users, AD, BI, ETL, profiling, modeling

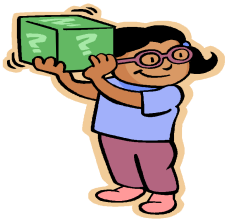
O
- Operational metadata
  - Information about application runtime
    - Frequency, record counts, component by component analysis and other statistics
  - Audience: Operations, management and business users

Literally, "*data about data*" that *describes* your company's *information* from *multiple perspectives.*

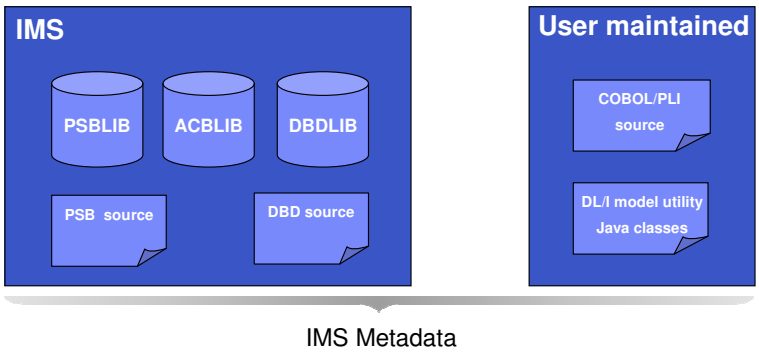
There are various types of "metadata".

The IMS catalog feature supports IMS Technical Metadata.

Before the IMS Catalog



- Databases partially defined in the IMS DBD
  - Only key/searchable fields needed by applications
  - Remaining segment data is not defined
- Remaining database definition is within Applications
  - COBOL COPYBOOKs and PL/I INCLUDEs map all the segment data
  - Applications can have different mappings for one segment



Prior to the implementation of the IMS catalog, metadata about all of the data contained in an IMS database is not defined within the DBD, only segment key fields and search fields are defined. To understand the data in a particular segment, the COBOL COPYBOOK or the PL/I INCLUDE must be reviewed. Sometimes, even the application code must be investigated to truly understand the database data.

With the IMS Catalog

- Database and program resources defined to an IMS system and relevant application information can be stored as metadata in an IMS catalog
  - Databases, fields, segments, data types, mappings and more …
- The IMS catalog is updated when you create, alter or delete IMS resource or application information
- Updates to the IMS catalog are done only via integrated IMS processes
  - Catalog Populate Utility (initial catalog load and member update)
  - PSBGEN, DBDGEN and ACBGEN
- Catalog metadata can be used to:
  - Enhance understanding of the data
  - Improve consistency of the data
  - Improve impact analysis of data
  - Improve development productivity
  - Improve data governance
- Key component of the overall IMS strategy:
  - Simplification
  - Integration

The IMS catalog contains information about IMS program resources, database resources that an IMS system controls, and relevant application metadata that an IMS application controls. This will include all programand database-related information defined to the IMS database system including databases, fields, segments, data types, etc., Changes made to any of these resources such as when you create, alter, or delete any IMS resource information will be reflected in the catalog. The only way to update the IMS catalog is through integrated IMS processes:

- Catalog Populate Utility
- PSBGEN
- DBDGEN
- ACBGEN
- Dynamic Database Change – Database Alter

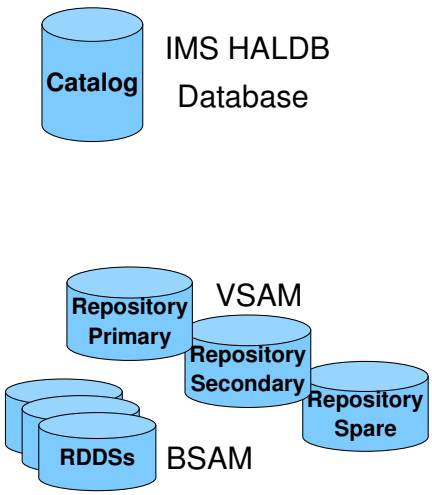
Any changes to COPYBOOK or INCLUDE members will require DBDGEN and ACBGEN processes.

As part of IMS's growth strategy, the IMS catalog is a key component which will play an important role in:

- Simplification
- Integration

Types of IMS Technical Metadata and Storage Method

- DB
  - PSB/DBD resources
    - Database structure definitions
    - Physical database definitions
    - Segment definitions
    - Field definitions
  - Application
    - Data types
    - Application defined fields
    - Encodings
    - Redefines
    - User defined types
    - Structures
- TM
  - MODBLKS resources
    - Program definitions
    - Transaction definitions
  - MFS FORMAT resources
  - Application
    - Input/output message definitions



Metadata exists for both IMS databases and transactions. It describes information that can be obtained from DBDs, PSBs, application COBOL copybooks or PL/I include files, and MODBLKS resources (GEN or DRD). A known requirement exists to be able to store MFS FORMAT and Application I/P and O/P messages. This is under consideration for a possible future enhancement.

The decision was made to store the database metadata in an IMS Catalog database, not the existing DRD Repository. Transactional metadata, on the other hand, is stored in the DRD Repository. Today, this metadata describes MODBLKS resources.

The IMS Catalog Database

- Contains metadata related to an IMS system's databases & programs
  - DBDs and PSBs and Application info
- IMS PHIDAM/OSAM HALDB database
  - Defined with 4 DSGs (Data Set Groups)
- Has one Secondary Index
- Unique feature
  - DBRC use is optional for the IMS Catalog HALDB database
    - **ONLY** HALDB that isn't required to be defined to DBRC
  - IMS can manage allocation/creation of catalog database data sets
    - Uses parameters in the "CATALOG" section of DFSDFxxx PROCLIB member

Optional DBRC usage is provided for ease of catalog database management across test systems.

The IMS Catalog Database is the only HALDB database for which DBRC is optional - it does not have to be defined in the RECONS.

IMS can manage the allocation/creation of catalog database data sets.
User must define the HALDB structure in the RECONs or in a definitional data set. The user must define recovery related options in the RECONs.

The IMS Catalog Database

- IMS provides DBD and PSB source code for the Catalog database
- IMS provides object code for the Catalog DBDs and PSBs
- PHIDAM DBD reserved name is DFSCD000
- PSINDEX DBD reserved name is DFSCX000
  - Used to connect DBDs to PSBs that reference them
- PSBs provided to load, read and update the Catalog database
  - DFSCPL00 is used for initial load process
    - Used by the Catalog Populate Utility
  - DFSCP000 (COBOL/HLASM), DFSCP002 (PL/I), and DFSCP003 (PASCAL) are used for read access
  - DFSCP001 is used for update access
    - Used by ACBGEN and Catalog Populate Utility
- Default catalog PCB is DFSCAT00
- ACBGEN required for all catalog PSBs

Source code for the catalog DBD and related PSBs are provided.

Load modules for the catalog DBD and related PSBs are also provided.

Users must generate the corresponding ACBLIB members for the PSBs before enabling, populating and using the IMS catalog.

The IMS Catalog Database

- Catalog database management is required
  - Review/adjust database buffer pool definitions
  - Perform routine management and maintenance on the Catalog database
    - Image Copy, Pointer Checker, Reorg, etc…
  - Catalog database will need to be REORG'd
    - If Catalog database is defined to the RECONs ➔
      - *HALDB OLR non-disruptive reorganization is supported*
    - If Catalog database is not defined to the RECONs ➔
      - *HALDB OLR can't be supported and a reorg utility must be employed*
- Support with existing backup and recovery procedures
  - Image copy, recovery, backout utilities etc…
  - If Catalog database is not defined to the RECONs ➔ recovery is limited
    - Same as non-registered, full function database recovery procedure

Existing recovery procedures that are used today for Image copy, recovery, backout utilities etc… can be used with the IMS Catalog Database. But, without DBRC enabled, Catalog database recovery is limited and the procedure will be similar to that used for a non-registered, full function database.

DBA management of the Catalog database is required. Users will need to verify that properly sized buffers exist in or are added to the IMS DB bufferpool definitions in the DFSVSAMP and DFSVSMxx PROCLIB members.

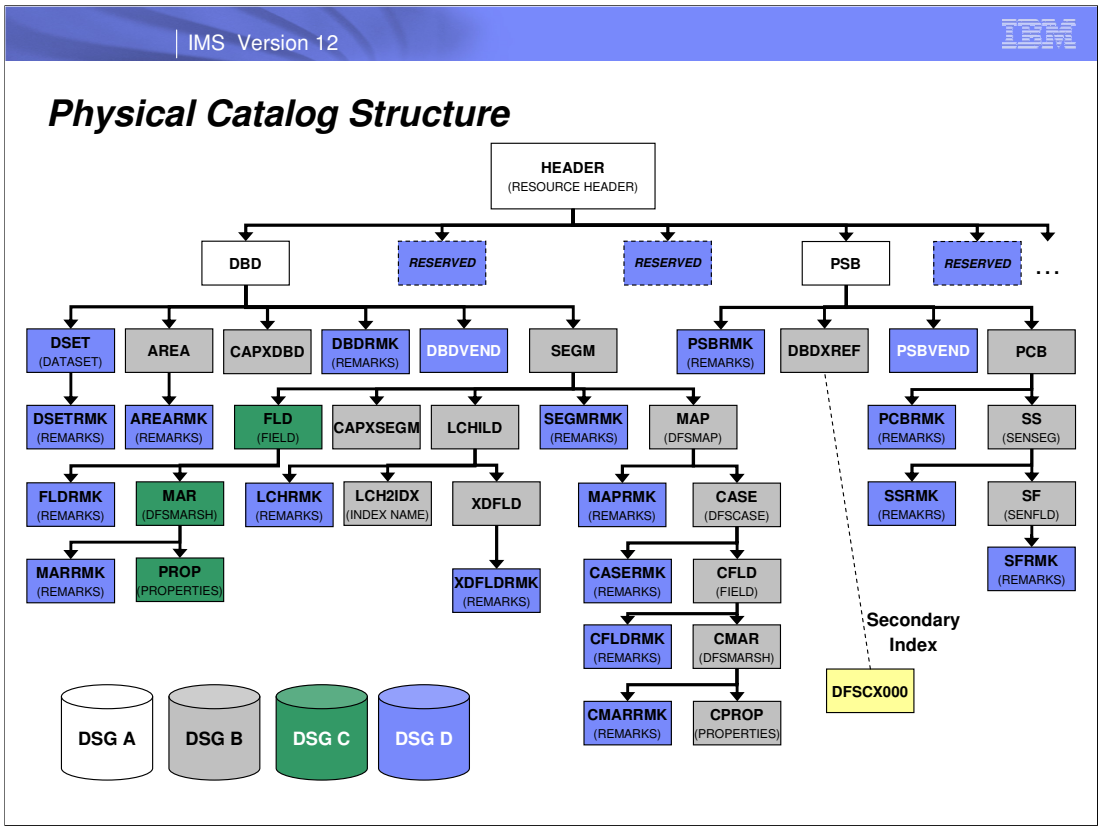
The database can also become disorganized and need reorganization. HALDB OLR (Online Reorganization) allows non-disruptive reorganization for the Catalog database. But if the Catalog database has not been defined in the RECONs, a batch reorganization utility must be used.

The IMS Catalog Database

- Root segment of the Catalog database is a generic resource header
  - Indicates the type of resource ➔ DBD or PSB
    - A dependent of the Root and it's Children are a complete DBD or PSB
      - Multiple versions of a specific resource are supported
        - Most resources are differentiated by their ACBGEN timestamp
        - Logical DBDs and GSAM DBs are differentiated by their DBDGEN timestamp
- Catalog database segments typically correspond to macro statements in the DBD and PSB source
- One segment at the first Child level under both the DBD and PSB segments is available for vendor/customer use
  - DBDVEND
  - PSBVEND

Let's look at the IMS Catalog database physical structure...

See the IMS Catalog database hierarchy on the next slide.



There is a segment available for ISV use at the first child level under both the DBD and PSB segments: DBDVEND and PSBVEND. Actually, anyone can use these segments!

There is a secondary index segment (DFSCX000) which points to the DBDXREF segment. This secondary index could be used to cross reference DBDs to PSBs that reference them.

There are also some new DBD macros used to define extended database info/metadata.

The Catalog database is comprised of four Data Set Groups (DSGs). The DBD definition specifies which segments reside in which DSGs.

IMS Catalog Database Restriction

WARNING:

Coexistence with previous IMS releases is not supported !

Access to the IMS Catalog Database from any IMS subsystem, program, utility or client executing or utilizing an IMS release earlier than IMS 12 is not supported !

- For Example:
  - Access to the Catalog database from an IMS 10 or IMS 11 online subsystem is not supported
  - Access to the Catalog database from the IMS Explorer via an IMS 10 or IMS 11 subsystem is not supported
  - Access to the Catalog database from an IMS 10 or IMS 11 DFSDDLT0 utility job is not supported

IMS Catalog Enablement

IMS Catalog Enablement

- Add catalog DBDs and PSBs to your DBDLIB, PSBLIB & ACBLIB
  - Copy DBD and PSB object code from SDFSRESL to your DBDLIB and PSBLIB

```
//CPYCMEM EXEC PGM=IEBCOPY
//SDFSRESL DD  DSN=SDFSRESL,DISP=SHR
//DBDLIB   DD  DSN=MYIMS.DBDLIB,DISP=OLD
//PSBLIB   DD  DSN=MYIMS.PSBLIB,DISP=OLD
//SYSIN    DD  *
       COPY   OUTDD=DBDLIB,INDD=((SDFSRESL,R)),LIST=YES
       SELECT MEMBER=(DFSCD000,DFSCX000)
       COPY   OUTDD=PSBLIB,INDD=((SDFSRESL,R)),LIST=YES
       SELECT MEMBER=(DFSCPL00,DFSCP000,DFSCP001,DFSCP002,DFSCP003)
```

  - ACBGEN the catalog DBD and PSB resources into your ACBLIB

```
      //CATACB  EXEC PGM=DFSRRC00,PARM='UPB'
      //STEPLIB  DD  DSN=SDFSRESL,DISP=SHR
      //DFSRESLB DD  DSN=SDFSRESL,DISP=SHR
      //IMS      DD  DSN=MYIMS.DBDLIB,DISP=SHR
      //         DD  DSN=MYIMS.DBDLIB,DISP=SHR
      //IMSACB   DD  DSN=IMS.ACBLIB,DISP=OLD
      //SYSIN    DD  *
               BUILD PSB=(DFSCPL00)
               BUILD PSB=(DFSCP001)
               BUILD PSB=(DFSCP000)
```

N.B. The MODBLKS resources for the Catalog databases and programs do not need to be defined !

SAMPLE JCL and Control Cards

- Copy the DBDs and PSBs for the IMS catalog from the IMS.SDFSRESL data set to the IMS.DBDLIB and IMS.PSBLIB data sets. The catalog DBDs are called DFSCD000 and DFSCX000. The PSBs are called DFSCPL00, DFSCP000, and DFSCP001, DFSCP002, DFSCP003.

- Run the ACB Maintenance utility to generate the ACBs for the IMS catalog.

- Activate the ACB library that contains the IMS catalog ACBs.

IMS Catalog Enablement

- Modify DFSDFxxx PROCLIB Member
  - New CATALOG section(s) for catalog related parameters
    - Single section format <SECTION=CATALOG>
    - Multiple section format <SECTION=CATALOG***imsid***>
      - Multiple IMS systems sharing one DFSDFxxx PROCLIB member
      - ***imsid*** suffix must be a four character IMS ID
  - CATALOG section parameters
    - CATALOG=**N** | Y
      - Catalog is disabled or enabled
      - If enabled, IMS automatically creates catalog DDIR & PDIRs at IMS startup
    - ALIAS=DFSC | xxxx (**no default value**)
      - Specifies any 1-4 alphanumeric value used as a Catalog database name prefix
      - Enables use of non-shared, Catalog databases within an IMSplex
        - Use in a data sharing environment where each IMS has its own Catalog database and all are registered in a single set of RECONs
        - At runtime, the alias Catalog database names are dynamically replaced with internal database names **DFSC**D000 and **DFSC**X000
      - For standalone IMS system – use "**DFSC**" which is the standard Catalog database name prefix ➔ **DFSC**D000 and **DFSC**X000

Code the CATALOG section(s) of the DFSDF*xxx* member in the PROCLIB data set. The Catalog Definition user exit routine (DFS3CDX0) is provided as an alternative option for batch processing environments that do not use the DFSDF*xxx* member of the PROCLIB data set.

The CATALOG and CATALOGxxxx sections of the DFSDFxxx member of the PROCLIB data set both use the same parameter list and syntax. For a single IMS system, always use the CATALOG section. In a multi-system environment, all IMS systems sharing this DFSDFxxx member use the options configured in the CATALOG section unless a CATALOGxxxx section is included for that specific IMS system. A DFSDFxxx member can contain any number of CATALOGxxxx sections, but only one CATALOG section.

When you add a CATALOGxxxx section, replace "xxxx" with the IMSID of an IMS system that uses this DFSDFxxx member.

CATALOG= Specifies if the IMS catalog is enabled or disabled. NO means the IMS catalog is disabled. This value is the default and is used if no CATALOG section is specified. YES means the IMS catalog is enabled.

ALIAS=*xxxx* Specifies a 1- to 4-character alphanumeric name prefix that is used to address a catalog database. References to the alias name are dynamically replaced with the internal catalog database and catalog secondary index names (DFSCD000 and DFSCX000) at runtime. There is no default value, but DFSC is the recommended value, which is the standard catalog name prefix.

IMS Catalog Enablement

- DFSDFxxx PROCLIB Member
  - CATALOG section parameters (continued)
    - Information used by Catalog Populate Utility to automatically allocate the Catalog database data sets
      - DATACLAS
        - Optional data class for SMS managed data sets
      - MGMTCLAS
        - Optional management class for SMS managed data sets
      - STORCLAS
        - Required storage class for SMS managed data sets
      - IXVOLSER
        - Volume serial number for primary and secondary catalog indices
        - Required for non-SMS managed data sets
      - SPACEALLOC
        - Free space % (0 to 9999) added to the IMS-computed size of the primary & secondary data set allocations
      - SMSVOLCT
        - Number of volumes (1-20) created by the Catalog Populate utility for SMS-managed

STORCLAS=*xxxxxxxx* Storage class for automatically generated, SMS-managed catalog data sets. Required for SMS-managed catalog data sets.

DATACLAS=*xxxxxxxx* Data class for automatically generated, SMS-managed catalog data sets.

MGMTCLAS=*xxxxxxxx* Management class for automatically generated, SMS-managed catalog data sets.

IXVOLSER=*xxxxxx* Volume serial number for all primary and secondary catalog indexes. Important: This parameter is required when the catalog data sets are not managed by SMS.

SMSVOLCT=*nn* Number of volumes created by the Catalog Populate utility for use by SMS-managed data sets. The valid range for this value is 1-20. The default is 1.

SPACEALLOC=(PRIMARY=*nnnn* SECONDARY=*nnnn*) This value is a percentage that is added to the IMS-computed size of the primary and secondary catalog data sets. The default for the primary dataset is 500% and the default for the secondary data set is 50%.

IMS Catalog Enablement

- DFSDFxxx PROCLIB Member
  - CATALOG section parameters (continued)
    - RETENTION=(MAX=**2** | nnn) or RETENTION=(PERIOD=**0** | nnn)
      - Specifies retention schedule for metadata in the IMS catalog
      - By default IMS keeps only two copies of the DBD or PSB in catalog
      - MAX=**2** | nnn
        - Maximum number of versions of a DBD or PSB to be stored before they are replaced first-in first-out
      - PERIOD=**0** | nnn
        - Maximum number of days a version of a DBD or PSB is to be stored before it can be replaced
        - Metadata versions older than the specified retention period are not automatically deleted, but available for removal when new version of metadata is added
        - Default value of “0” disables this feature

RETENTION= This optional statement specifies the retention schedule for metadata in the IMS catalog. By default, the IMS catalog stores 2 versions of DBD and PSB metadata that is currently being used in the IMS system. If a previous version exists in the database, it is replaced when a newer version is created. Specify different values to maintain multiple versions of IMS catalog metadata.

MAX=*nnn* Specifies the maximum number of versions of the catalog metadata to store in the IMS catalog. When the number of versions exceeds this value, the oldest version (based on the ACBGEN timestamp) is removed in favor of the newest version. This value is a decimal value from 1 to 255. The default is 2.

PERIOD=*nnn* Specifies the retention period of metadata in the IMS catalog in days. When the age of a version of the catalog metadata exceeds this value, it is removed from the IMS catalog the next time a new version of the catalog metadata is added. This parameter does not automatically delete all versions of the catalog metadata that are older than the specified retention period. It makes them available for removal only when new versions of the same DBD or PSB are added to the IMS catalog. This value is a decimal from 0 to 999. Specifying 0 (the default) disables this feature.

IMS Catalog Enablement

- DFSDFxxx PROCLIB Member
  - Enabling the IMS catalog for IMS batch processing
    - Specify DFSDFxxx member on job EXEC parm
      - Requires JCL change to implement
    - User Exit Routine
      - Optional Catalog Definition user exit routine, DFS3CDX0
      - Alternative to specifying DFSDFxxx member through job JCL
      - Available if users cannot or choose not to modify job JCL

The DFSDFxxx PROCLIB member must be present for any IMS subsystem to determine if the IMS catalog is enabled.

For IMS Batch Processing:

1) IMS job JCL can be modified to specify the DFSDFxxx member on thejob exec parm

2) A Catalog Definition user exit routine (DFS3CDX0) is provided as an alternative option for batch processing environments when you do not want to specify the DFSDF*xxx* member on the job exec parm

Today, batch use of the new IMS catalog metadata may not be considered a real hot or important feature, unless you are already running Java batch processes which access IMS databases. But, it is possible that in the future the catalog could be totally integrated with all IMS database processing.

IMS Catalog Enablement

- Definition of the HALDB structure
  - Partitioning of the catalog is the users responsibility
    - Minimum of 1 partition is required
    - Last partition must be able to contain the highest-key PSB record
    - Catalog HALDB uses the high-key selection method
      - No use of Partition Selection Exit is allowed
  - Catalog Database Definition
    - For systems that use DBRC
      - Catalog database can be defined to the RECONs with the DBRC utility and commands
    - For systems that do not use DBRC
      - Catalog database must be defined to the Catalog Partition Definition data set using the Catalog Partition Definition Data Set utility, DFS3UCD0
    - If an ALIAS is used in the CATALOG sections of the DFSDFxxx member, each alias Catalog database must be defined

Define the HALDB master and partitions of the IMS catalog. This information must be defined either to DBRC or with the Catalog Partition Definition Data Set utility (DFS3UCD0) for IMS systems that do not use DBRC.

The IMS™ catalog is a HALDB (high availability large database) database, a partitioned, IMS full-function database type. Prior to loading records into an IMS catalog, you must define the partitions to IMS. The HALDB partition definitions for the IMS catalog are typically registered in the RECON data set by DBRC. DBRC provides support for database recovery, data sharing, automatically generating JCL control statements, log and data set management, and other database tasks. The IMS catalog, however, does not have to be registered with DBRC. You can define the IMS catalog database partitions with the IMS catalog partition definition utility (DFS3UCD0) instead of using DBRC. The DFS3UCD0 utility stores the catalog partition information in the catalog partition definition data set. This option is not available for other HALDBs. When the IMS catalog partition definition data set is used, the IMS catalog is not registered with DBRC. The database protection support provided by DBRC, including backup and recovery support, is unavailable to the IMS catalog when a IMS catalog partition definition data set is used.

The HALDB Partition Data Set Initialization utility (DFSUPNT0) is not compatible with an unregistered catalog database. The Catalog Populate utility (DFS3PU000) provides analogous support for registered and unregistered catalog databases.

The IMS catalog database does not support a HALDB Partition Selection exit routine (DFSPSE00). The catalog database uses the high-key selection method exclusively.

IMS Catalog Enablement

- Using DBRC DSPURX00 utility and commands to define the Catalog database to the RECONs

```
//DEFCAT EXEC PGM=DSPURX00
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//SYSPRINT DD SYSOUT=*
//IMS DD DSN=IMS.DBDLIB,DISP=SHR
//SYSIN DD *
 INIT.DB DBD(DFSCD000) TYPHALDB SHARELVL(3)
 INIT.PART DBD(DFSCD000) PART(DFSCD01) -
    DSNPREFX(dsnprefix.DFSCD000) -
    BLOCKSZE(4096) -
    KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF')
 INIT.DB DBD(DFSCX000) TYPHALDB SHARELVL(3)
 INIT.PART DBD(DFSCX000) PART(DFSCX01) -
    DSNPREFX(dsnprefix.DFSCX000) -
    KEYSTRNG(X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF') /*
```

  - If using a catalog alias prefix, replace ***DFSC*** in the database and partition names for the catalog and the catalog secondary index with the four character ALIAS name prefix
  - You might need to define multiple alias name databases to the RECONs

The DBRC utility can be used to define the Catalog database structure to the RECONs.

If alias prefix names were defined in the DFSDFxxx member, those alias named catalog databases must be defined.

IMS Catalog Enablement

- Using the Catalog Partition Definition Data Set utility, DFS3UCD0, to define the Catalog database (for systems that do not use DBRC)

```
//S1 EXEC PGM=DFS3UCD0,REGION=0M
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS.SDFSRESL,DISP=SHR
//DFSHDBSC DD DSN=...,DISP=
//SYSPRINT DD SYSOUT=*
//IMS DD DSN=IMS.DBDLIB,DISP=SHR
//SYSIN DD *
 HALDB=(NAME=DFSCD000)
 PART=(NAME=DFSCD000,PART=partitionname,
         DSNPREFX=dsnprefix,
         KEYSTRNG=keystring)
 HALDB=(NAME=DFSCX000)
 PART=(NAME=DFSCX000,PART=partitionname,
         DSNPREFX=dsnprefix,
         KEYSTHEX=FFFFFFFFFFFFFFFF) /*
```

Unregistered Catalog database

  - Catalog Partition Definition data set is populated with the information specified in the HALDB and PART control cards
    - RECON-like information for catalog database partition definition and structure
  - The name DFSCD000 in the HALDB and PART statements contains the default catalog prefix ***DFSC***. If your catalog uses an alias name prefix, substitute it in the JCL

If you choose not to use DBRC for the Catalog database, you will have to use the Catalog Partition Definition Data Set Utility, DFS3UCD0, to build/populate a Catalog Partition Definition data set with the HALDB database structure information – HALDB, PART, DSN Prefix, KEY info.

DFSHDBSC is the DDNAME for the Catalog Partition Definition data set.

IMS Catalog Enablement

- After Catalog database is defined in Catalog Partition Definition Data Set
  - Identify unregistered Catalog database names
    - UNREGCATLG parameter in the DATABASE section of the DFSDF*xxx* member

```
/*****************************************************************/
/* Database Section */
/*****************************************************************/
<SECTION=DATABASE>
UNREGCATLG=(DFSCD000,DFSCX000) /* Unregistered IMS catalog DB     */
/*****************************************************************/
/*                                                               */
/*****************************************************************/
```

  - If using an alias name prefix, replace ***DFSC*** in the UNREGCATLG database names with the four character alias name prefix
  - Limitations of using an unregistered Catalog database
    - NO IMS Data Sharing support
    - NO OLR support
    - NO partition definition change support
      - User must rebuild catalog partitions
    - Manual recovery required for unregistered Catalog databases

You might need to define multiple alias name databases as UNREGCATLG.

IMS Catalog Enablement

- After Catalog database is defined in Catalog Partition Definition Data Set
  - Create a new DFSMDA dynamic allocation member for the Catalog Partition Definition data set

```
//DYNALOC JOB
//*
//STEP EXEC IMSDALOC
//SYSIN DD *
   DFSMDA TYPE=INITIAL
   DFSMDA TYPE=CATDBDEF,DSNAME=dsn
   DFSMDA TYPE=FINAL
   END
/*
```

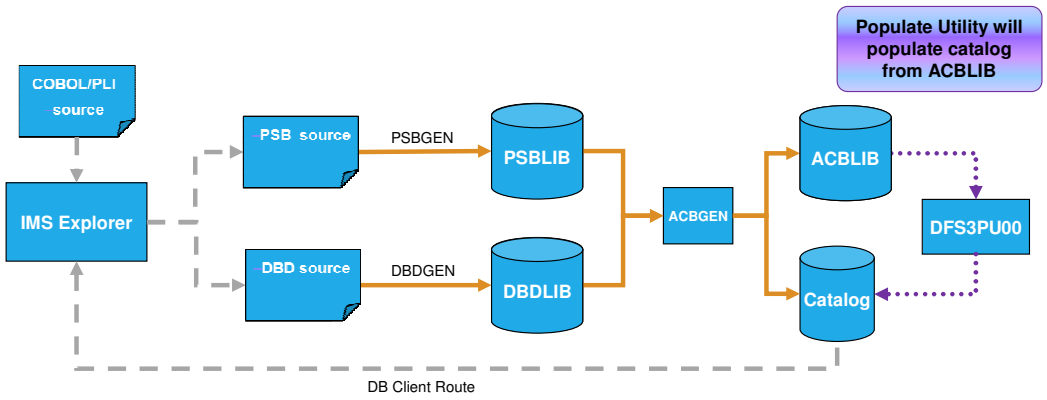
  - ***dsn*** is the name of the Catalog Partition Definition data set
    - Catalog Partition Definition data set was allocated in the DFS3UCD0 utility on the DFSHDBSC DD
  - Dynamically allocate Catalog Partition Definition data set in any IMS job step

You can create a Dynamic Allocation member for the Catalog Partition Definition data set.

DFSHDBSC is the DDNAME used for the Catalog Partition Definition data set.

IMS Catalog Lifecycle

IMS Catalog Lifecycle



- ACBGEN will populate ACBLIB and catalog in same UOW
  - Populates ACBLIB with *standard* ACB info and *extended* info
  - Populates the catalog with *extended* info
- Key points
  - Only way to update catalog is via the Populate Utility or ACBGEN process
  - Extended info is acquired via the IMS Explorer
  - Extended info stored in ACBLIB members for recoverability

Steps:

1. Initially populate of the Catalog with the Catalog Populate Utility (Only done once)
2. Generate new DBD and PSB source with the IMS Explorer to incorporate application metadata
3. Propagate the PSB and DBD source in the same manner as before into ACBLIB

Note that the Catalog and ACBLIB are updated in the same unit of work.

Key point…to change one field in one segment you will need an ACBGEN….tradeoff is that it is completely trusted

Key point…to get the good stuff into the catalog requires a manual process of importing COBOL/PLI into the Explorer

IMS Catalog Creation

- Populate the IMS Catalog
  - Load the IMS Catalog using IMS Catalog Populate utility, DFS3PU00
    - Each ACB member is decoded, converted to catalog format, loaded into the catalog
  - Reads ACBLIB, DBDLIB and PSBLIB datasets as input
    - Data sets can be concatenated but only first occurrence of an ACB member is used
    - DBDLIB needed for Logical databases and GSAM databases
    - PSBLIB needed to determine which GSAM databases go into the catalog
  - Catalog database DBD and PSB segments will have a version and contain a timestamp that matches the ACB member timestamp
    - Used to associate an ACB member with a catalog member
    - Timestamp exceptions
      - DBDGEN timestamp for Logical and GSAM DBs
      - PSBGEN timestamp for GSAM only PSBs

Use the IMS Catalog Populate utility , DFS3PU00), to load or insert records into the IMS catalog database data sets. The DFS3PU00 utility creates the catalog records from the ACB members in one or more ACB libraries and, depending on your database types, the associated DBD and PSB members in DBD libraries and PSB libraries. The records contain metadata for your application programs and databases. The utility can run in a DL/I region or, if updating an existing IMS catalog, can run in a BMP region.

The DFS3PU00 utility requires access to the following data sets:

- The IMS.PROCLIB data set that contains the DFSDFxxx member that enables the IMS catalog and defines the alias name of the IMS catalog
- One or more IMS.ACBLIB data sets
- If the IMS Catalog supports logically related databases, the IMS.DBDLIB data set
- If the IMS Catalog supports GSAM databases, the IMS.DBDLIB data set and the IMS.PSBLIB data set

If any of the database data sets do not exist, the DFS3PU00 utility creates them automatically:

> The DFSCD000 database data sets which are:

- The primary index data set
- The indirect list data set (ILDS)
- Four data set group data sets for the segments of the IMS catalog

> The DFSCX000 secondary index data set

The DFS3PU00 utility calculates the size of the database data sets based on both the size of the ACB library and the expansion percentages that you can specify in the DFSDFxxx PROCLIB member.

The DFSDFxxx member is also where you specify the volume serial number for the VSAM key-sequenced data sets or the SMS storage class, data class, and management class for all data sets

IMS Catalog Creation

- IMS Catalog Populate utility (DFS3PU00)
  - Can run as a typical IMS Batch or BMP job
    - Requires IMS logs for backout / recovery
    - Requires IRLM if catalog is shared and catalog active in an IMS subsystem
      - Business as usual for data sharing
    - BMP mode allows for updates to catalog in non-data sharing environment
    - Requires DBRC if catalog is defined in the RECON
    - If using Catalog Partition Definition Data Set
      - Users responsible to ensure online catalog access has ceased
      - Business as usual for non-registered full function DB
  - Can also be used to insert additional records to an existing Catalog database

The DFS3PU00 utility loads the IMS catalog by using the IMS-provided PSB DFSCPL00, which includes a PCB defined with PROCOPT=L.

```
//UPDTCAT EXEC PGM=DFS3PU00,
// PARM=(DLI,DFS3PU00,DFSCPL00,,,,,,,,,,Y,N,,,,,,,,,,,'DFSDF=001')
```

Running the DFS3PU00 utility with the PSB DFSCPL00 deletes any existing records in the IMS catalog!

IMS Catalog Creation

- Load the IMS Catalog using the new ACB Generation and Catalog Populate utility, DFS3UACB
  - Generates the ACB library members and loads the IMS catalog metadata in the same job step
  - Not recommended if ACBLIB is already valid
    - No need to recreate the ACBLIB, just populate the catalog

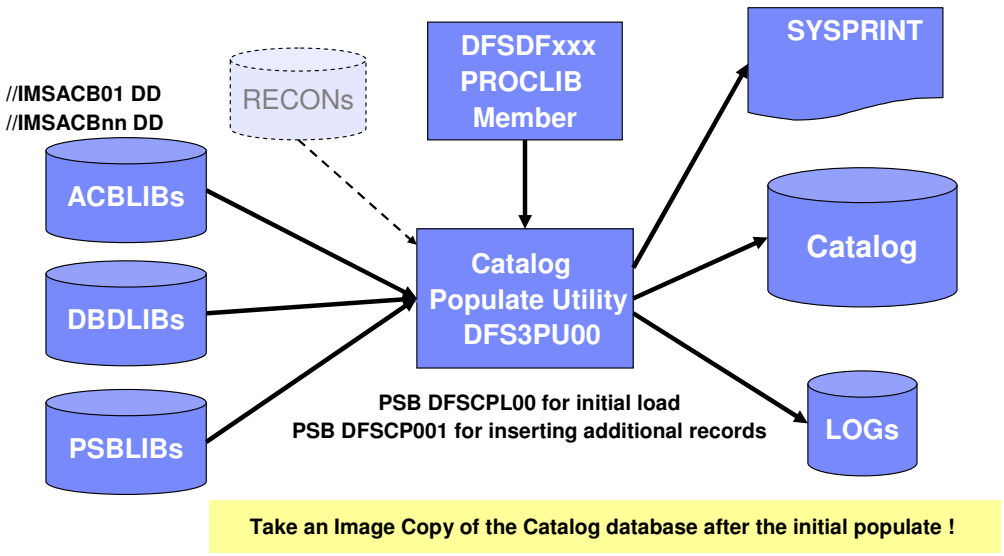
You can generate ACB members and populate the IMS catalog in a single step by using the ACB Generation and Catalog Populate utility (DFS3UACB).

The DFS3UACB utility can be used to initially load the IMS catalog or to add records to an existing IMS catalog. When the DFS3UACB utility populates the IMS catalog, the utility stores the ACB members in the IMS.ACBLIB data set and internally calls the IMS Catalog Populate utility (DFS3PU00) to load or update the records in the IMS catalog in the same unit of work (UOW).

For the ACB generation phase of the utility execution, the operation requirements are the same as those of the ACB Maintenance utility.

It is recommended that you use the Catalog Populate Utility for the initial load of the IMS catalog. If you are starting with a good ACBLIB, there is no reason to reGEN all of the ACBs before populating the catalog with metadata.

Catalog Populate Utility – DFS3PU00



Use the IMS Catalog Populate utility, DFS3PU00, to load or insert records into the IMS catalog database data sets. The DFS3PU00 utility creates the catalog records from the ACB members in one or more ACB libraries and, depending on your database types, the associated DBD and PSB members in DBD libraries and PSB libraries. The records contain metadata for your application programs and databases. The utility can run in a DL/I region or, if updating an existing IMS catalog, can run in a BMP region.

The amount of space that the utility allocates for the data sets is based on the ACB library members and the values specified on SPACEALLOC parameter in the catalog section of the DFSDFxxx PROCLIB member.

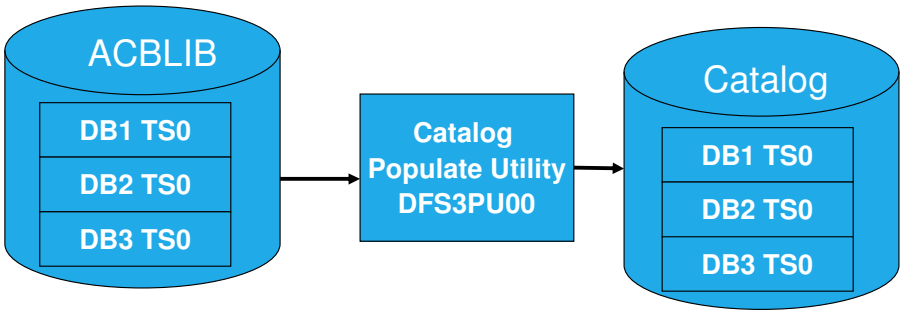
The output of the DFS3PU00 utility includes a report that contains statistics about the record segments that are loaded into the IMS catalog. The report includes information about the number and types of segments, as well as an estimate of the amount of DASD storage that each data set group of the IMS catalog will require. **If you need to know how much DASD storage the IMS catalog data sets will use before they are created, you can run the DFS3PU00 utility without populating the IMS catalog to generate only the statistics report.**

As an alternative to running the DFS3PU00 utility, you can populate the IMS catalog by using the ACB Generation and Catalog Populate utility (DFS3UACB). The DFS3UACB utility generates the ACB libraries for your applications and databases and then populates the IMS catalog, both in the same job step.

When the IMS catalog is registered with DBRC, you are required to create an image copy of the IMS catalog after an initial load of the IMS catalog.

When the IMS catalog is not registered with DBRC, IMS cannot require an image copy after an initial load; however, if an image copy is not created after an initial load, the only way to recover the catalog is to reload it.

IMS Catalog Members After Populate Utility



- There are 3 members in an ACBLIB
- Run the Catalog Populate Utility, DFS3PU00
- The catalog members will have a timestamp TS0
- This is the ACB member timestamp

564

The populate utility reads the ACB members and creates the corresponding metadata entries in the catalog.

The members of the IMS.ACBLIB data set must be generated by a version 12 or later IMS system. The members of the IMS.DBDLIB and IMS.PSBLIB can be generated by IMS systems earlier than IMS version 12; however, the resulting records in the IMS catalog will not include certain metadata segments.

You can include multiple ACB libraries as input to the utilities that populate the IMS catalog. The data sets for the input ACB libraries can be specified in separate ddnames, in a concatenation for a single ddname, or using a combination of these. When duplicate ACB members are encountered, the duplicates are not processed.

The DFS3PU00 utility considers ACB members to be duplicates of each other if both their ACB member names and timestamps are identical. The DFS3PU00 utility does not load data into the catalog from an ACB member that is either a duplicate of another ACB member read as input from another ACB library or of an ACB member from which data has previously been loaded into the IMS catalog. If an input ACB member has the same name as a previously read ACB member, but has a different timestamp, the data from both ACB members is loaded into the IMS catalog.

Note: When the input data sets for multiple ACB libraries are concatenated in a single DD statement, ACB members that have the same member name as a member from a data set earlier in the concatenation are completely ignored, even if the timestamps are different. Only the first instance of an ACB member with a given name in the list of concatenated data sets is passed to the utility as input. Consequently, if you need to load the data from ACB members that have the same name but different timestamps into the IMS catalog, specify the applicable ACB libraries by using separate DD statements.

Adding IMS Catalog Metadata

- New ACB Generation and Catalog Populate utility, DFS3UACB
  - Replaces existing ACBGEN Utility, DFSUACB0, if IMS catalog enabled
  - Generate ACBLIB member and create catalog metadata in a ***single job step***
    - Phase 1 - ACBGEN
      - DBDLIB and PSBLIB members used as input
      - Validation is unchanged
      - ACB member is written to ACBLIB with new ACBGEN timestamp
    - Phase 2 – IMS catalog update
      - Generated ACB is decoded, converted to catalog format, loaded into the catalog
        - DBD and PSB metadata created and inserted
        - Corresponding ACB member timestamp saved as timestamp in catalog DBD and PSB segments
        - ensures validity and consistency of ACBLIB and catalog
- New ACBGEN, DFS3UACB, and new Catalog Populate, DFS3PU00, utilities are the only updaters of the IMS catalog
  - IMS online and IMS batch regions will never update catalog data
  - IMS online and IMS batch regions will only retrieve data from the catalog

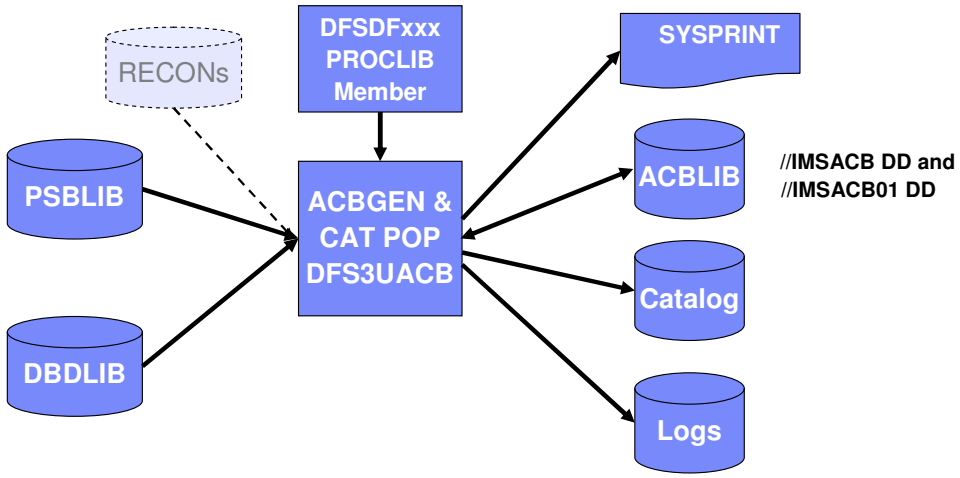
Use the ACB Generation and Catalog Populate utility (DFS3UACB) to generate ACB members in an IMS.ACBLIB data set and create the corresponding metadata records in the IMS™ Catalog in a single job step. Populating the IMS catalog in the same job step as the generation of the ACB members ensures that the proper ACB library is used to populate the IMS catalog.

If you do not need to generate the ACB library members and populate the IMS catalog in the same job step, you can run the ACB maintenance utility and then the IMS Catalog Populate utility (DFS3PU00) in separate jobs or job steps.

The DFS3UACB utility can populate the catalog in load mode or in update mode. When load mode is used, any existing records in the IMS catalog are discarded.

If you are updating an existing IMS catalog, consider creating an image copy of the IMS catalog data sets. If the IMS catalog is registered with DBRC, you can use the DBRC command GENJCL.IC to backup the catalog. If you have defined the IMS catalog in an IMS Catalog partition definition data set, you must use standard image copy JCL.

IMS Catalog Additions Via New ACBGEN & Populate Utility



- New integrated ACBGEN process includes update to the IMS catalog
- DFSDFxxx PROCLIB member has the catalog information
- DFSMDA member used to dynamically allocate the catalog datasets

566

ACBGEN has been enhanced to create the catalog metadata at the same time we create the ACB member. This keeps them in sync. Both will have the timestamp of the ACBGEN.

The catalog dataset name is in the DFSDFxxx PROCLIB member. Using this, we can allocate the catalog datasets.

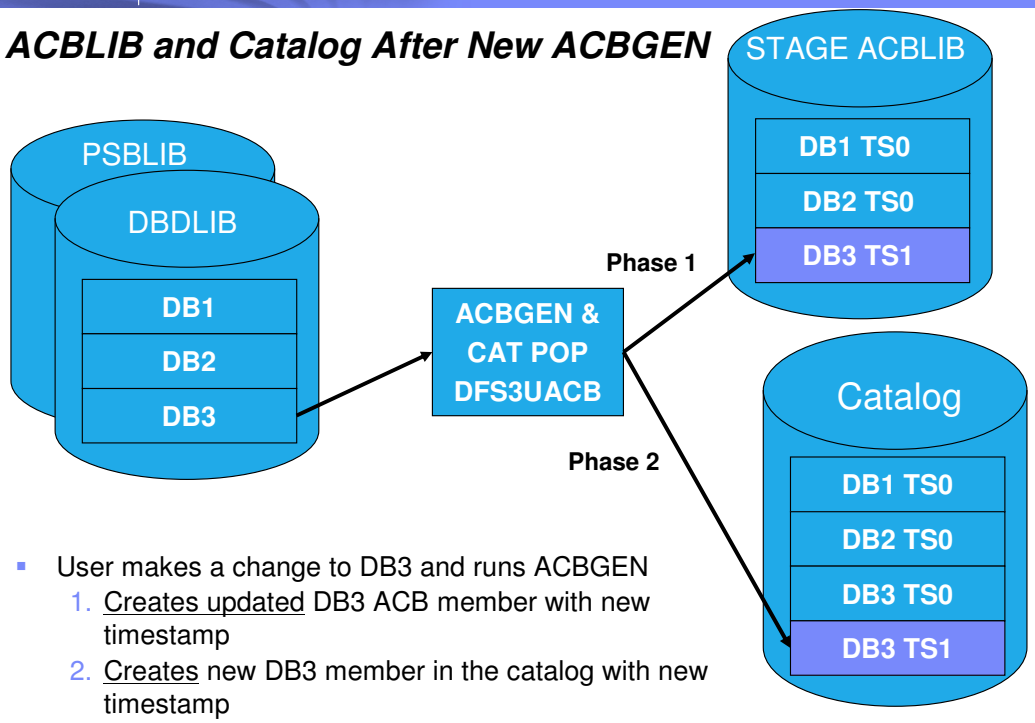
The ACBGEN process will invoke the ACB decoder and put the member back into a format that the builder will create records to be inserted into the catalog database.

//**IMSACB DD** Defines the IMS.ACBLIB data set.

Restriction: This data set is modified and cannot be shared with other jobs.

//**IMSACB01 DD** Defines an ACB library data set that contains the ACB members that are used to populate the IMS catalog. This DD statement is required. This DD statement must specify the same data set defined in the IMSACB DD statement. To ensure that the same data set is referenced, code this DD statement with an asterisk as the high-level qualifier, as shown in the example:
//IMSACB01 DD DSN=*.ACBLIB,DISP=OLD

ACBLIB and Catalog After New ACBGEN



- User makes a change to DB3 and runs ACBGEN
  1. Creates updated DB3 ACB member with new timestamp
  2. Creates new DB3 member in the catalog with new timestamp

567

In this example, there are 3 members in the ACBLIB; DB1, DB2 & DB3. There is a timestamp in each ACBLIB member and the timestamp is from when each ACB member was created. In the example it is represented by TS0.

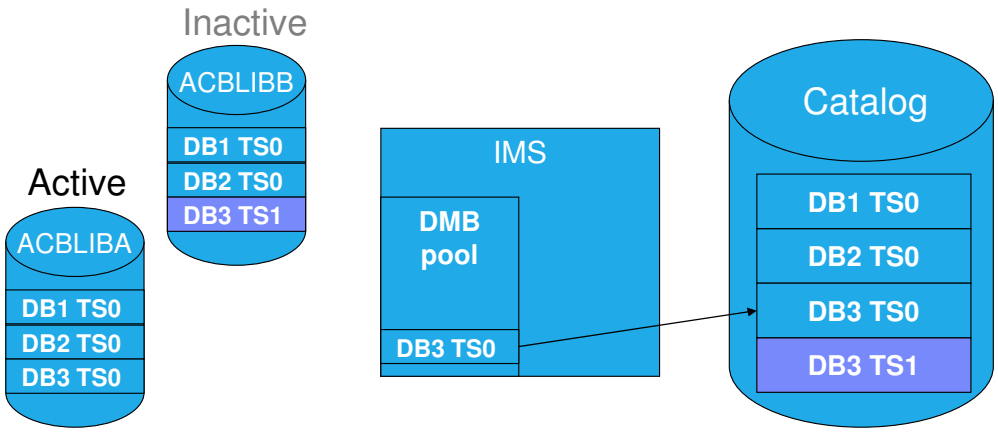
In this scenario, the application programmer makes changes to DB3. The DBA does a DBDGEN followed by an ACBGEN. The ACBGEN will create an updated DB3 member in the ACBLIB. The new ACBLIB member will have a new time stamp, TS1.

As part of the ACBGEN, a member is created in the catalog that corresponds to the updated ACB member. This catalog member will have the same timestamp as the ACB member. This allows us to keep them in sync. The member in the catalog has the name DB3 and the timestamp TS1.

The changes and testing are done in the user test environment.

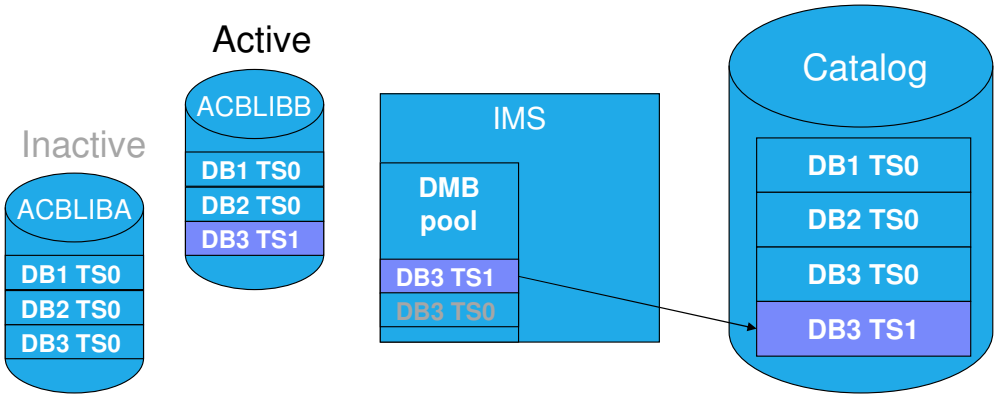
Catalog Member Timestamps

- Java application request is made to read DB3
  1. IMS determines active DB3 ACB member has timestamp TS0
  2. Internal DL/I call issued to retrieve member DB3 from IMS catalog
  3. IMS retrieves catalog member DB3 with timestamp TS0



Catalog Member Timestamps

- Initiate OLC to switch from ACBLIBA to ACBLIBB
  - Activates DB3 ACB with timestamp TS1
- Java application request is made to read DB3
  1. IMS determines active DB3 ACB member has timestamp TS1
  2. Internal DL/I call issued to retrieve member DB3 from IMS catalog
  3. IMS retrieves catalog member DB3 with timestamp TS1



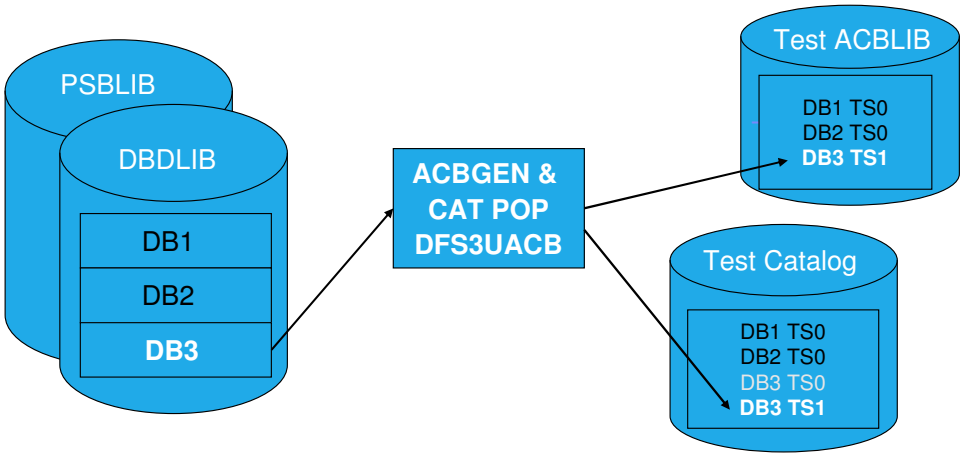
IMS Catalog and ACBLIB Migrations

Catalog and ACBLIB Migration from Test to Production

- Migration process for changed resources does not change significantly
- Use new ACB Generation and Catalog Populate utility, DFS3UACB
  - DFS3UACB replaces existing ACBGEN Utility DFSUACB0
- When copying ACB members from staging to inactive ACBLIB:
  - Use new IMS Catalog Copy Utility, DFS3CCU0
    - DFS3CCU0 replaces Online Change Copy utility DFSUOCU0
    - Ensures that the catalog and ACBLIB are kept in sync

Testing Environment - Step I

- Developer or DBA makes a change to DB3 and runs ACBGEN
  1. Creates updated DB3 ACB member with new timestamp
  2. Creates new DB3 member in the catalog with new timestamp



572

This example starts out with an application programmer making changes to DB3. The changes and testing is done in the user test environment.

In this example, there are 3 members in the ACBLIB; DB1, DB2 & DB3. There is a timestamp in each ACBLIB member of when the ACB member was created. In the example it is represented by TS0.

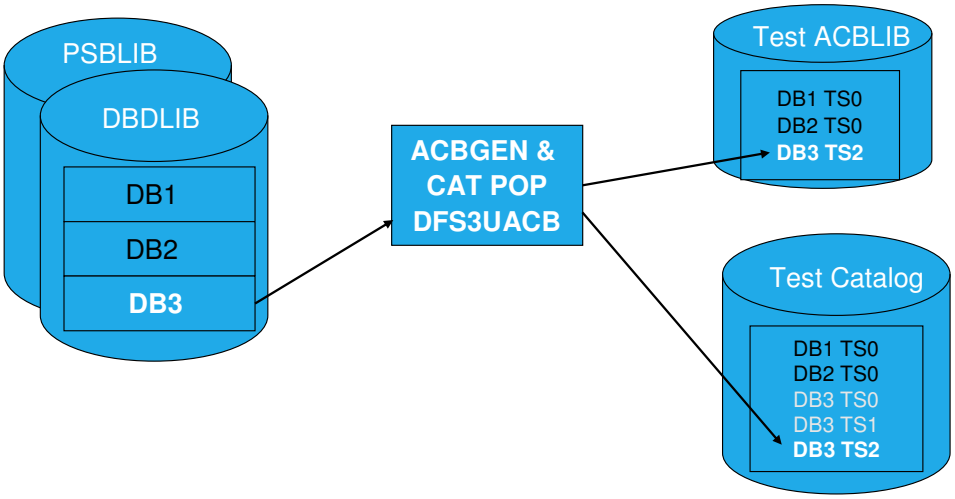
In this scenario, the application programmer updates DB3. They do a DBDGEN followed by an ACBGEN. The ACBGEN will create an updated member in the ACBLIB for DB3. It will have a new time stamp that is part of the ACB member. This is identified in the example as TS1.

As part of the ACBGEN, we will also create a member in the catalog that corresponds to that ACB member. This catalog member will have the same timestamp as the ACB member. This allows us to keep them in sync.

The member in the catalog has the name DB3 and the timestamp TS1. It will also contain a user version value. In V12 it's just expressed as V0. In future releases as we support user versioning, it will be the actual version of the ACB.

Testing Environment - Step II

- Developer or DBA makes another change to DB3 and runs ACBGEN again
  1. Creates updated DB3 ACB member with new timestamp
  2. Creates another new DB3 member in the catalog with new timestamp

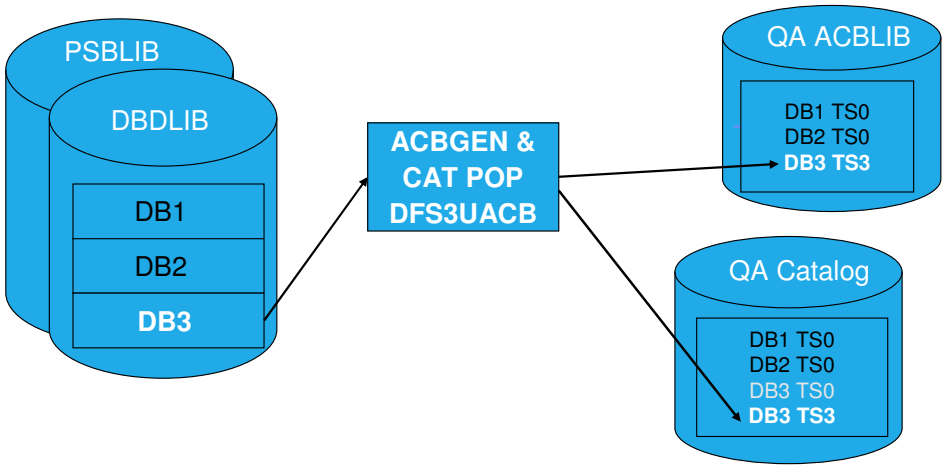


573

The application programmer has to make another change to DB3. Runs DBDGEN and another ACBGEN. This will create an updated member in the ACBLIB. This new member has timestamp TS2. There is a new member created in the catalog that corresponds to the new ACB member. It also has the new timestamp TS3. The version stays the same. The old member with timestamp TS1 still stays in the catalog.

QA Environment - Step III

- Developer testing is completed, ready for User testing in the QA system
  1. Run ACBGEN for DB3 into the QA ACBLIB and the QA catalog
  2. Creates updated DB3 ACB member with new timestamp
  3. Creates new DB3 member in the catalog with new timestamp



574

When testing is complete, the application will be migrated to the QA environment. The user does an ACBGEN from the DBDLIB and creates an updated member with new timestamp TS3. A new member is created in the catalog with timestamp TS3.

Catalog and ACBLIB Migration from QA to Production

- Ready to propagate corresponding ACBLIB members and IMS catalog members between QA and PROD IMS environments
  - Need to copy ACBLIB members and corresponding catalog metadata
    - New Catalog Copy Utility, DFS3CCU0 (recommended method)
    - Catalog Populate Utility, DFS3PU00
- Catalog Copy Utility, DFS3CCU0, can be used for …
  - Migration of resource changes from one IMS environment to another
  - IMS System cloning
  - Maintaining Disaster Recovery system
- Catalog Copy Utility, DFS3CCU0, is a two step process
  - Export step
    - creates an export data set based on catalog member from ACBLIB
  - Import step
    - uses export data set to write to both IMS catalog and ACBLIB
    - single step updates ACBLIB member and catalog metadata

Catalog Copy Utility - DFS3CCU0 Export Step

- Exports member(s) from an input IMS catalog
  - Runs as an IMS Batch or BMP job
- Export process
  - Unloads a single catalog member at a time
    - Cannot export different versions of the same catalog member
  - Uses ACBLIB to get timestamp for a catalog member export
    - If no ACBLIB provided → copy all the catalog members
    - If no ACBLIB member found → copy the catalog member anyway
  - Read catalog member looking for the ACBLIB timestamp match
  - Write catalog member to the export data set
- Input
  - ACBLIB and corresponding IMS catalog
- Output
  - Export data set containing unloaded catalog members
  - Similar format and size as an unload data set

Catalog Copy Utility – DFS3CCU0 Import Step

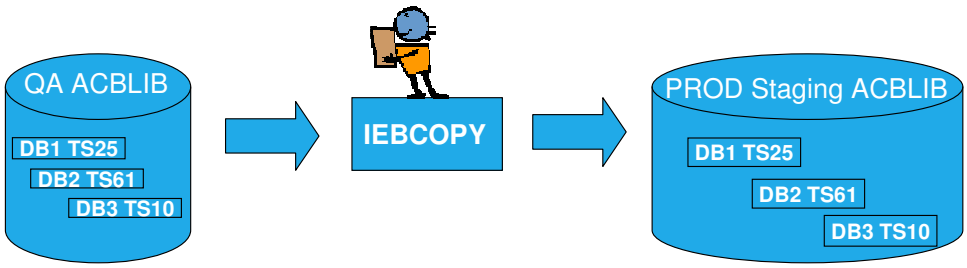
- Imports member(s) to an IMS catalog
  - Runs as an IMS Batch or BMP job
- Import process
  - Write catalog member from export dataset to the IMS catalog
  - Internal IEBCOPY of ACBLIB member that matches the catalog member timestamp
    - If no ACBLIB provided → skip the copy of all catalog members
    - If no matching ACBLIB member found → skip the catalog member copy
- Input
  - Input ACBLIB
  - Export data set containing unloaded catalog members
- Output
  - Output ACBLIB member and corresponding catalog member

Scenario: IMS Resource Migration

- Use the Catalog Copy Utility to migrate resource changes from one IMS environment to another
  - Example: Migrate IMS QA to IMS PROD environment
    - ACBGEN is done only in IMS QA environment
    - PROD Staging ACBLIB is clone of QA ACBLIB
      - same timestamps
    - ACBLIB members will determine which catalog members are copied to the production catalog
  - Process may vary slightly depending on method used for modifying IMS resources
    - OLC, GOLC, MOLC or IMS restart (if not using OLC)

Migration to Production Environment - Step IV

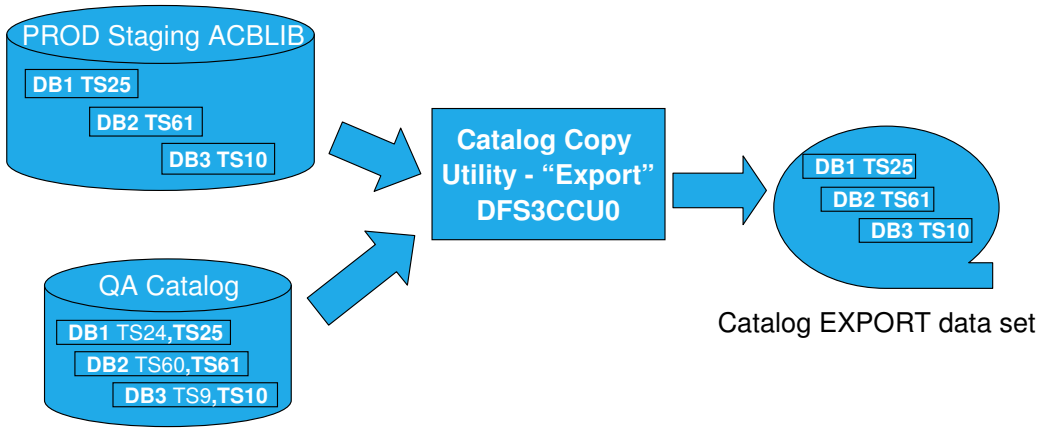
- User testing in the IMS QA system is completed, ready to migrate changes to the IMS PROD system
  1. Copy members from QA ACBLIB to PROD Staging ACBLIB



Testing on the QA libraries is done so the user is now ready to migrate the libraries to production. The QA staging ACBLIB is copied to the production ACBLIB using standard IEBCOPY utility.

Migration to the Production Environment - Step IV

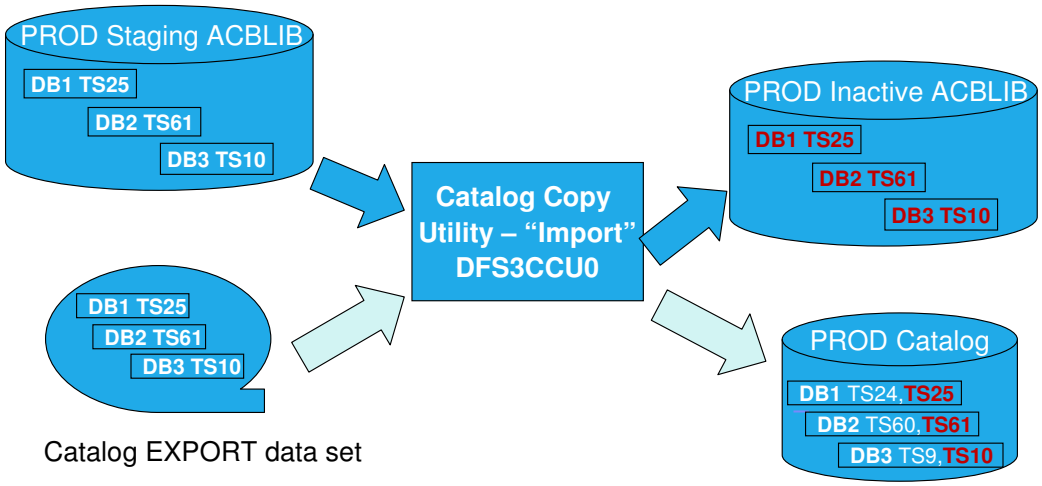
2. Export IMS QA catalog members that have a matching timestamp in PROD Staging ACBLIB



The staging ACBLIB is copied to the inactive ACBLIB to prepare for the online change. The existing online change copy utility is used.

Migration to the Production Environment - Step IV

3. Import IMS catalog members and ACBLIB members into IMS PROD system
   - Catalog members from export data set migrated to IMS PROD catalog
   - ACBLIB members with timestamps matching catalog export data set members migrated from PROD Staging ACBLIB to PROD Inactive ACBLIB

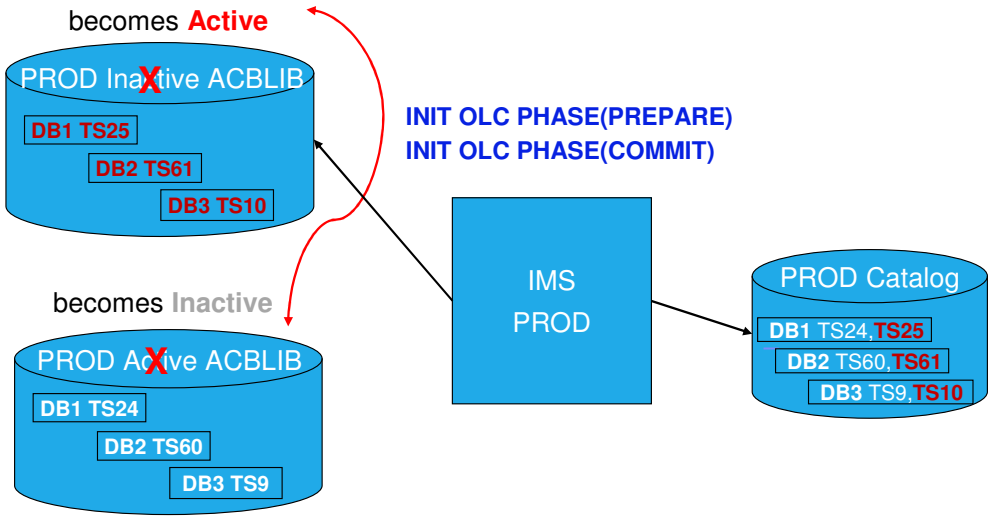


581

The staging ACBLIB is copied to the inactive ACBLIB to prepare for the online change. The existing online change copy utility is used.

Migration to Production Environment - Step IV

4. Operator issues OLC commands to bring the new ACB resources online
   - Inactive ACBLIB becomes active and active ACBLIB becomes inactive
   - Catalog has both active and inactive members



582

An online change is done to bring the new DB3 resource online

Supplement

IMSplex Catalog Sharing

IMS Catalog Sharing

- Catalog Database Data Sharing
  - One Catalog database can be defined and shared among IMS systems
  - Follows standard IMS data sharing protocols and procedures
  - Use standard processes for database management and monitoring
- Non-Shared Catalog database
  - A separate Catalog database can be defined for each IMS system
  - DFSDFxxx <SECTION=CATALOG> ALIAS parameter allows multiple Catalog databases to exist within same RECON
  - Catalog databases should not be shared if not registered with DBRC
  - Catalog Populate utility can be run as a BMP
    - Prevents a Catalog database outage when catalog updates needed

The Catalog database is a normal HALDB and follows all the standard data sharing protocols. Existing procedures and processes can be used for the shared IMS Catalog DB. Data sharing of a Catalog database is only supported on IMS 12 or higher subsystem.

To support environments that require more than one catalog database, IMS can perform catalog database name aliasing. The DFSDF*xxx* member of the IMS.PROCLIB data set referenced in the start-up JCL for the IMS system can contain a catalog database name alias. The alias replaces the prefix DFSC for both the catalog database and catalog secondary index.

If you are not data sharing, the ALIAS can be use to identify unique IMS catalog database names that can reside in the same RECONs. ALIAS=*xxxx* specifies a 1- to 4-character alphanumeric name prefix that is used to address the catalog database. References to the alias name are dynamically replaced with the internal catalog database and catalog secondary index names (DFSCD000 and DFSCX000) at runtime. The default is DFSC, which is the standard catalog name prefix. Any 4 bytes can be used…but you might want to use the 4 byte IMSID for ease of Catalog data base name recognition.

If not data sharing the IMS Catalog DB, the populate utility can be run as BMP. This will prevent a Catalog DB outage when you have catalog updates

Catalogs that are not registered with DBRC should not be shared.

For a standalone IMS system or for multiple IMS systems sharing a common catalog, use one DFSDFxxx member with one <SECTION=CATALOG> definitions.

For IMS systems not sharing a catalog, use one DFSDFxxx member with multiple <SECTION=CATALOGxxxx> definitions. Append the four byte IMS ID to “CATALOG” in the section heading: <SECTION=CATALOGIMS1> and <SECTION=CATALOGIMS2>

IMS Catalog Sharing

- DFSDFxxx PROCLIB Member
  - Can be shared or not shared among multiple IMS systems
  - New CATALOG section differentiates whether catalogs are shared or not shared
    - If only one definition for <SECTION=CATALOG>
      - One shared Catalog database
      - ALIAS=**DFSC** recommended
    - If multiple definitions for <SECTION=CATALOG***imsid***>
      - ***imsid*** differentiates that a Catalog database is not shared
      - Replace "***imsid***" in section heading with four byte IMS ID of the IMS system that uses this specific CATALOG

        <SECTION=CATALOG**IMS1**> or <SECTION=CATALOG**IMS2**>
      - Unique ALIAS=**xxxx** for each Catalog database
    - A standalone IMS system uses a DFSDFxxx member with one <SECTION=CATALOG> definition

The Catalog database is a normal HALDB and follows all the standard data sharing protocols. Existing procedures and processes can be used for the shared IMS Catalog DB.

ALIAS=*xxxx* specifies a 1- to 4-character alphanumeric name prefix that is used to address the catalog database. References to the alias name are dynamically replaced with the internal catalog database and catalog secondary index names (DFSCD000 and DFSCX000) at runtime. The default is DFSC, which is the standard catalog name prefix. Any 4 bytes can be used…but you might want to use the 4 byte IMSID for ease of Catalog data base name recognition.

If not data sharing the IMS Catalog DB, the populate utility can be run as BMP. This will prevent a Catalog DB outage when you have catalog updates

Catalogs that are not registered with DBRC should not be shared.

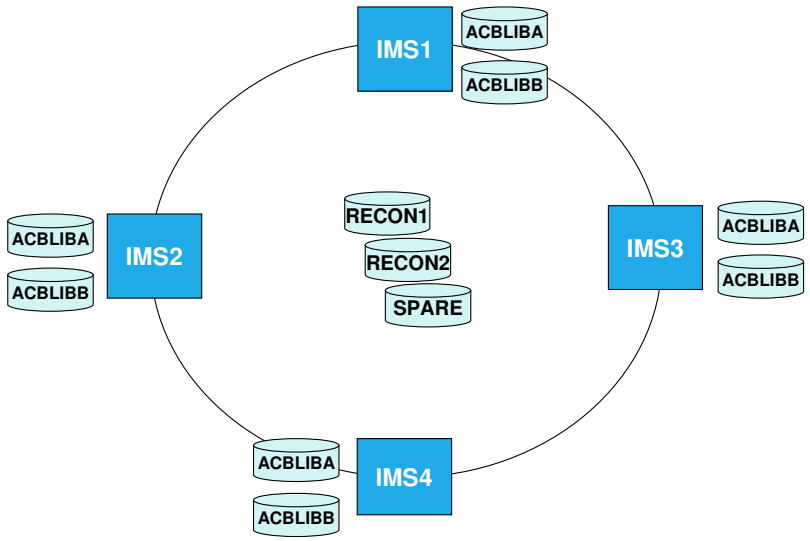
For a standalone IMS system or for multiple IMS systems sharing a common catalog, use one DFSDFxxx member with one <SECTION=CATALOG> definitions.

For IMS systems not sharing a catalog, use one DFSDFxxx member with multiple <SECTION=CATALOGxxxx> definitions. Append the four byte IMS ID to "CATALOG" in the section heading: <SECTION=CATALOGIMS1> and <SECTION=CATALOGIMS2>

IMS Catalog Sharing

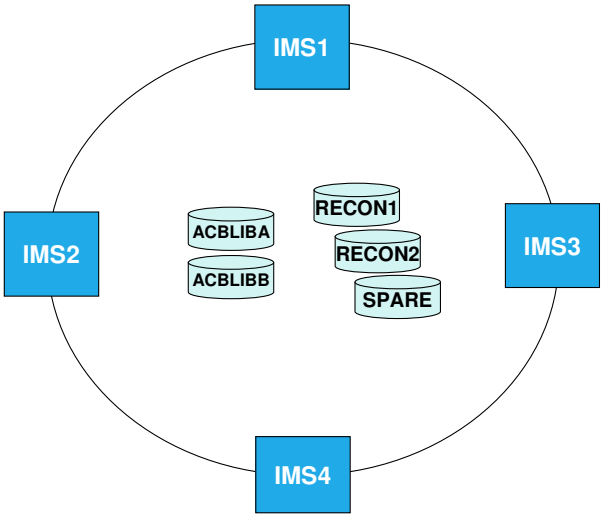
- Catalog ALIAS definition
  - Catalog database ALIAS names must be defined in the DBDLIB
    - Add ALIAS names to the Catalog database name list in your DBDLIB with the IMS Catalog ALIAS Names utility, DFS3ALI0
  - Alias name Catalog database must be defined in the RECON or the Catalog Partition Definition data set
    - Multiple Catalog databases may exist in the RECON
  - Alias name Catalog databases are not automatically maintained by IMS
    - User is responsible to keep Catalog databases in sync
    - Users with cloned ACBLIBs must have processes to keep ACBLIBs in sync during changes
      - Processes need to be modified to keep ACBLIBs and catalog in sync

Without an IMS Catalog
Multiple IMSes, each IMS has it's own cloned ACBLIBs



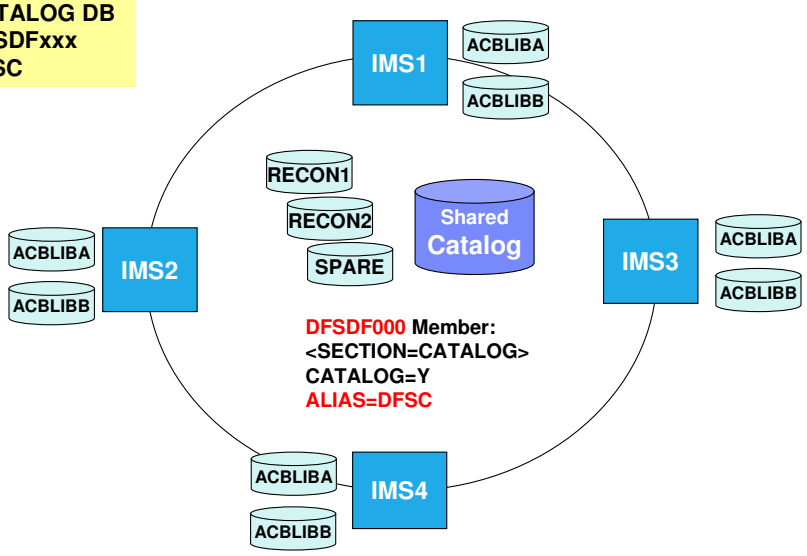
588

Without an IMS Catalog
Multiple IMSes, shared ACBLIBs



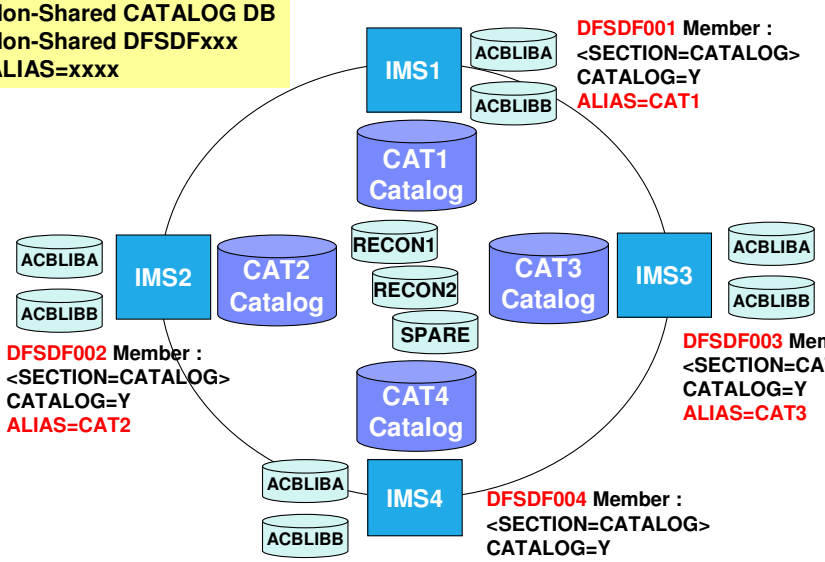
Multiple IMSes, Cloned ACBLIBs, Shared Catalog

Shared CATALOG DB
Shared DFSDFxxx
ALIAS=DFSC



Multiple IMSes, Cloned ACBLIBs, each IMS has it's own Catalog

Non-Shared CATALOG DB
Non-Shared DFSDFxxx
ALIAS=xxxx



DFSDF001 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT1

DFSDF002 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT2

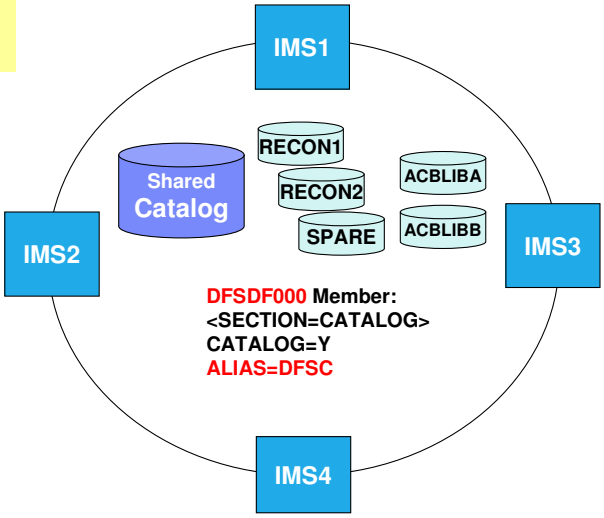
DFSDF003 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT3

DFSDF004 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT4

591

Multiple IMSes, Shared ACBLIBs, Shared Catalog

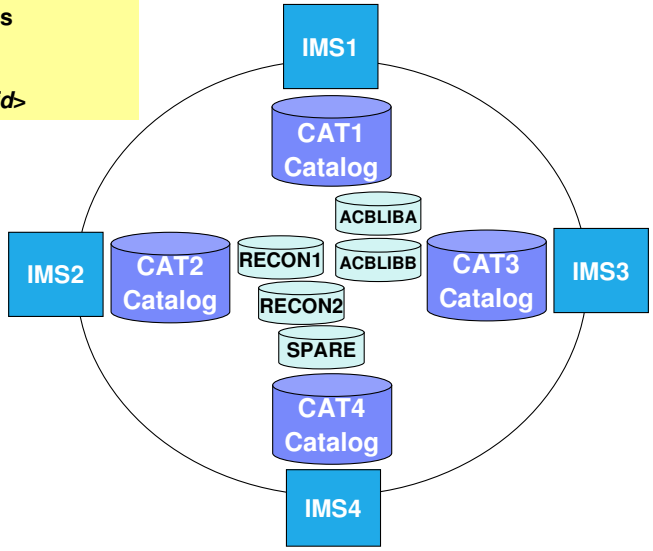
Shared CATALOG DB
Shared DFSDFxxx
ALIAS=DFSC



Multiple IMSes, Shared ACBLIBs, each IMS has it's own Catalog

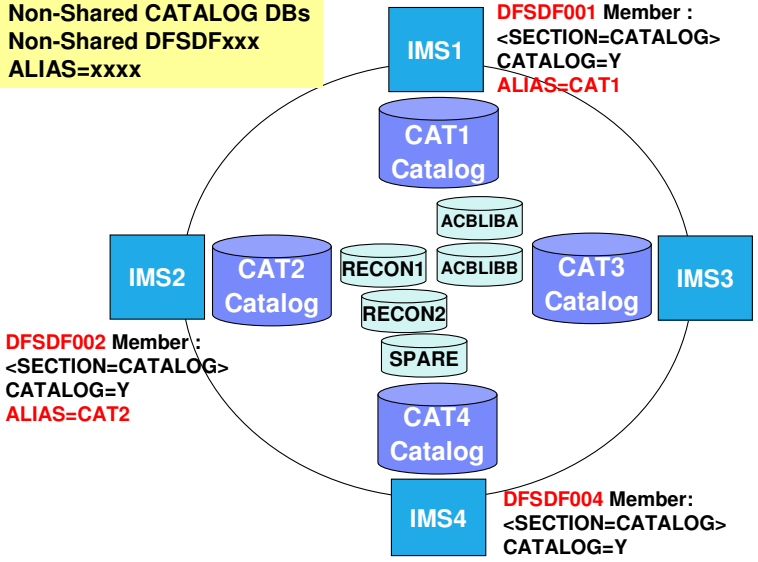
Non-Shared CATALOG DBs
Shared DFSDFxxx
Separate
<SECTION=CATALOG*imsid*>
ALIAS=xxxx
DFSDF000 Member:

<SECTION=CATALOGIMS1>
CATALOG=Y
ALIAS=CAT1
<SECTION=CATALOGIMS2>
CATALOG=Y
ALIAS=CAT2
<SECTION=CATALOGIMS3>
CATALOG=Y
ALIAS=CAT3
<SECTION=CATALOGIMS4>
CATALOG=Y
ALIAS=CAT4



Multiple IMSes, shared ACBLIBs, each IMS has it's own Catalog

Non-Shared CATALOG DBs
Non-Shared DFSDFxxx
ALIAS=xxxx



DFSDF001 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT1

DFSDF002 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT2

DFSDF003 Member :
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT3

DFSDF004 Member:
<SECTION=CATALOG>
CATALOG=Y
ALIAS=CAT4