IBM Software Group

IMS Version 9
Database and DBRC Enhancements

Hanne Nestinger

IBM Germany, Software Group

ON DEMAND BUSINESS

Session Objectives

- After this session you should be able to:
  - Explain the enhancements to IMS database in V9
  - Explain the enhancements to DBRC in IMS V9

Agenda: DB and DBRC Enhancements

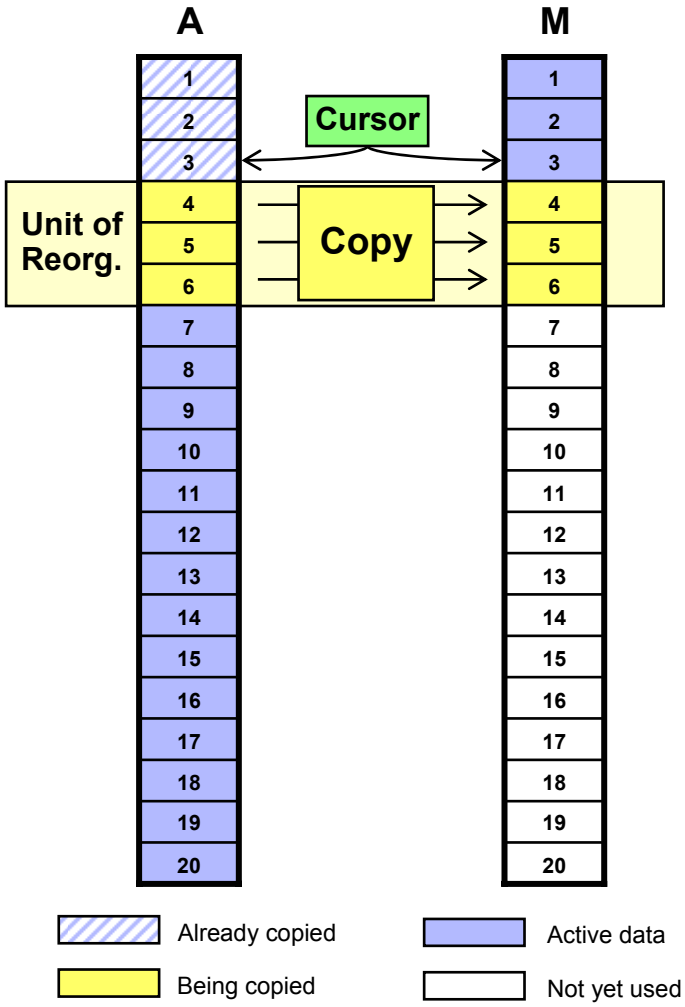
- Database Enhancements
  - HALDB Online Reorganization
  - HALDB Specific Partition Initialization
  - Image Copy Large Tape Block Sizes
  - IRLM 2.2
  - Multi-Area Structures for DEDB SVSO
  - Fast Path Area Open/Close Enhancements
  - XML DB
  - IMSplex Database Commands
  - PS EDI
  - Improved Message with Database Abends
- DBRC Enhancements
  - Command Authorization for /RMxxxx commands
  - More than 32K Database Registration
  - HALDB Dynamic Allocation by GENJCL.IC
  - DBRC Application Programming Interface (API)

HALDB Online Reorganization

- Absolutely no outage for reorganizations
  - Applications access all of the data during reorganizations without restriction
- Online reorganization technique
  - Writes new data sets
    - Dynamically allocates output data sets (optional)
    - Deletes input data sets when reorganization completes (optional)
    - Duplicate data sets
      - Only for partitions being reorganized
      - Only during the reorganization of the partitions
- Supports data sharing
  - Other IMS systems may read and update the partitions while they are reorged
  - Reorganization may be done in any data sharing IMS system

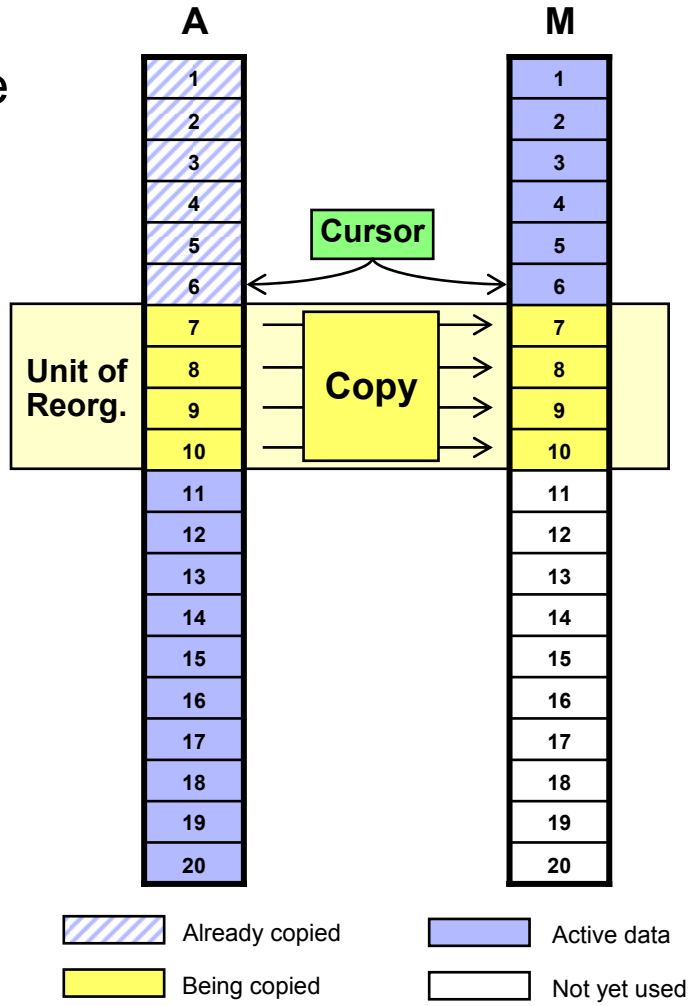
HALDB Online Reorganization

- Two active sets of data sets
  - Both are used during the reorg.
  - Records are copied to new data set
- 'Unit of Reorg'
  - Set of records being copied are one time
  - Records are locked during copy
    - Number of records in UOR is dynamically adjusted
    - Algorithm limits time taken, bytes copied, and locks held at any time
  - Cursor determines which data set contains active record



HALDB Online Reorganization

- Data set used is based on cursor value
  - Cursor on Record 6
  - Access Record 5:
    - Access from M data set
  - Access Record 14:
    - Access from A data set
  - Access Record 9:
    - Wait for lock,
      - then access from M data set



HALDB Specific Partition Initialization

- Background
  - HALDB partitions must be initialized before they may be used
- Previous capabilities of HALDB Partition Initialization Utility
  - Initialize partitions for a list of databases
    - Initializes only partitions with ‘partition initialization needed’ flag on

or

  - Initialize all partitions without regard to the ‘part. init. needed’ flag
- Added capabilities
  - Initialize a list of partitions
    - Initialize partitions without regard to the ‘part. init. needed’ flag
- Benefits
  - Improved flexibility
    - Especially important for testing
      - Eliminates need to turn on the ‘part. init. needed’ flag

Large Image Copy Block Size on Tape

- Tape block sizes > 32K allowed for image copies
  - Requires supporting hardware (e.g. 3590)
  - Invoked when BLKSIZE on Image Copy JCL DD statement
    - Not specified (system determined block size)
    - >32K
  - Supported by:
    - Image Copy utility (DFSUDMP0)
    - Database Recovery utility (DFSURDB0)
    - Database Recovery Facility tool (DRF)
- Benefits
  - Improved performance and greater tape capacity

IRLM 2.2

- 64-bit addressing support
  - Uses 64 bit addressing if z/Architecture and z/OS 1.3 or greater
    - Most lock control blocks are placed above the bar
      - Provides support for more concurrently held locks
      - Primarily required for DB2 systems with “abusive locking”
  - Uses 31-bit addressing if not z/Architecture or not z/OS 1.3 or greater
- Benefits
  - Greater concurrent lock capacity

IRLM 2.2

- IMS V9 includes IRLM 2.2
  - IRLM 2.1 is also shipped with IMS V9
- Compatibility
  - IRLM 2.1 may be used with IMS V7, V8, and V9
  - IRLM 2.2 may be used with IMS V7, V8, and V9
  - IRLM 2.1 and 2.2 may coexist in the same IMS data sharing group

Multi-Area Structures for DEDB Shared VSO

- IMS V9 allows multiple areas to share a CF structure
  - Previous releases required a CF structure for each shared area
  - System-managed duplexing may be used
    - Provides increased availability with simplified definitions and operation
  - Multi-area structures defined in DBRC
    - Areas assigned to the same structure in DBRC use the same buffer pool
- Benefits
  - Simplified structure management
  - Avoids potential z/OS limitation of 512 structures per sysplex

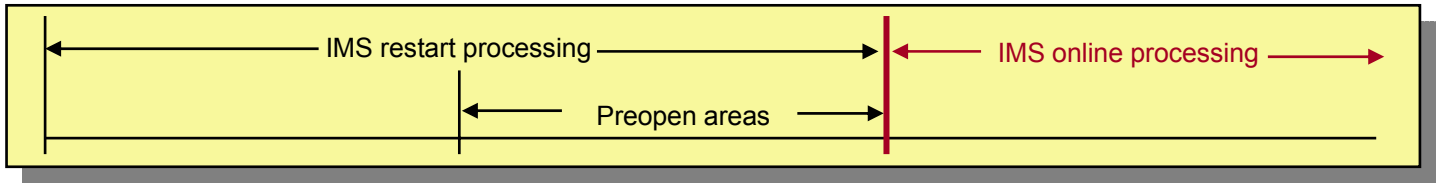
Fast Path Area Open and Close Enhancements

- New capabilities
  - Parallel open and close processing
  - Async open processing with online activity
    - Does not delay IMS restart completion
  - Options to (re)open areas after /ERE completes
    - Open only those with PREOPEN specified (like previous releases)
    - Open all of those open at time of termination
    - Open both those with PREOPEN and those open at time of termination
  - Restart and reopen of areas after IRLM reconnect
- Benefits
  - Faster restarts and terminations of IMS
  - Simplified operations

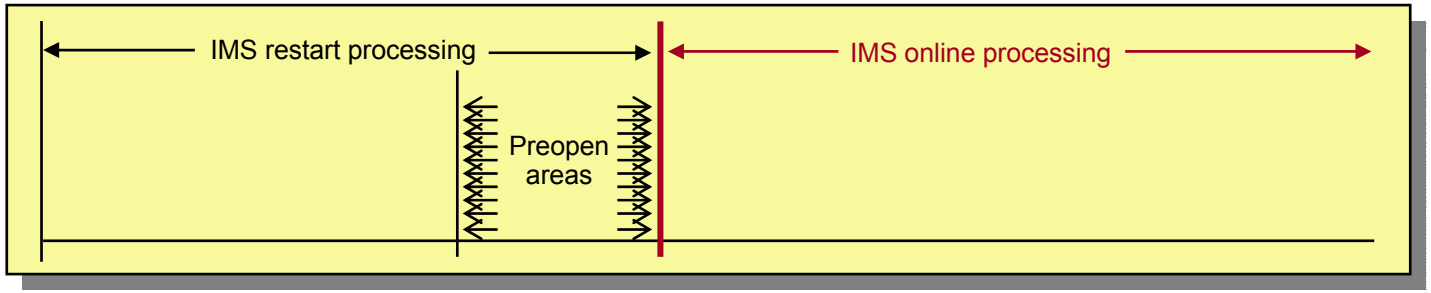
Fast Path Area Open and Close Enhancements

IMS Restart Processing for DEDB Areas

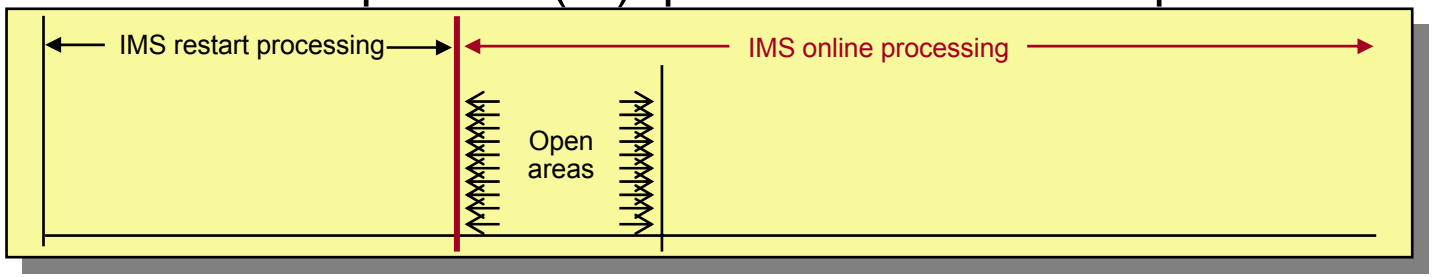
- Pre-IMS V9



- IMS V9 with option to open during restart



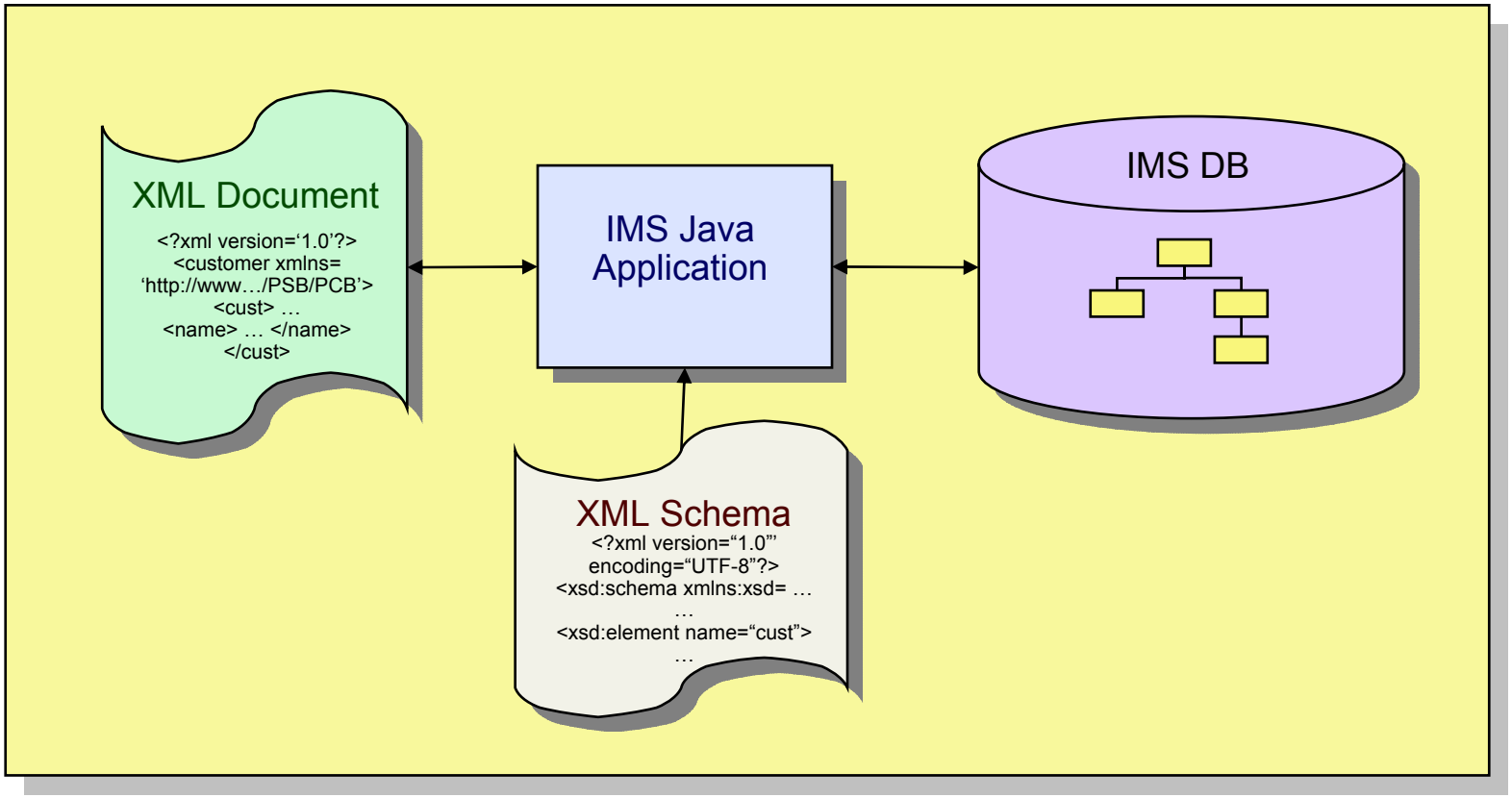
- IMS V9 with option to (re)open after restart completes



XML Database

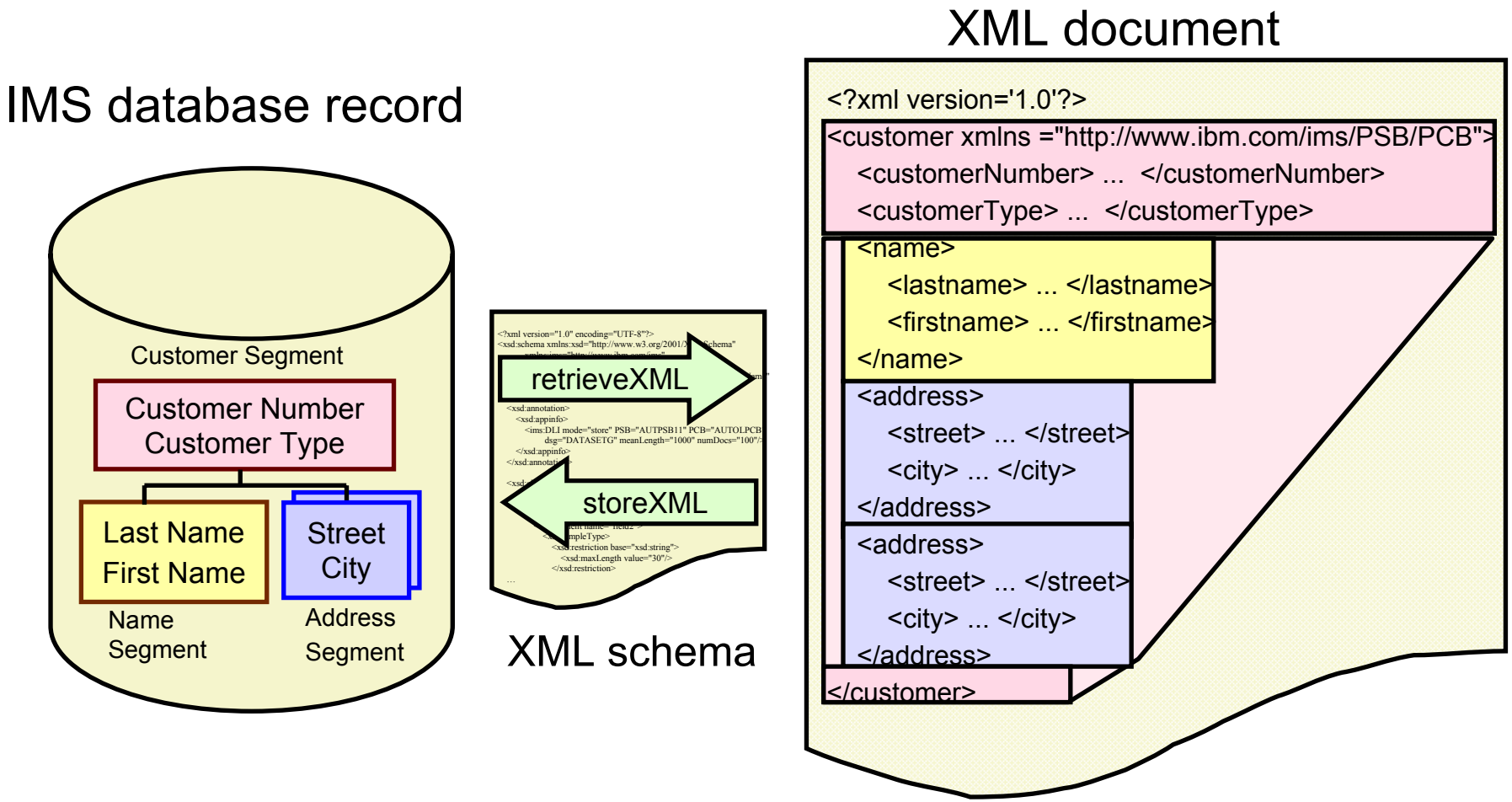
- Storage and retrieval of XML documents in IMS databases
  - Composition of XML documents from existing IMS databases
  - Creation of IMS segments from XML documents (decomposition)
  - Intact storage of XML documents (without decomposition)
  - Tooling assistance to define metadata for mappings
  - IMS Java application programming support
- Benefits
  - Easy exchange of data between IMS databases and XML documents
    - Existing IMS databases may be used to create XML documents
    - Existing applications are unaffected

XML Database



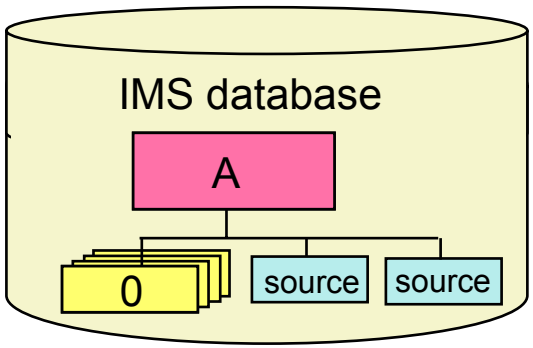
Decomposed Storage Example

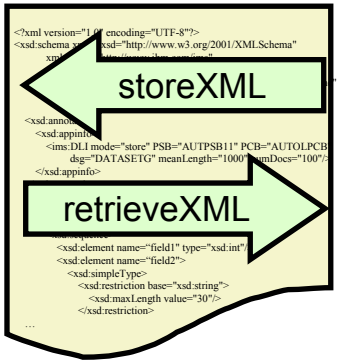
- Data is usable by IMS applications with no knowledge of XML



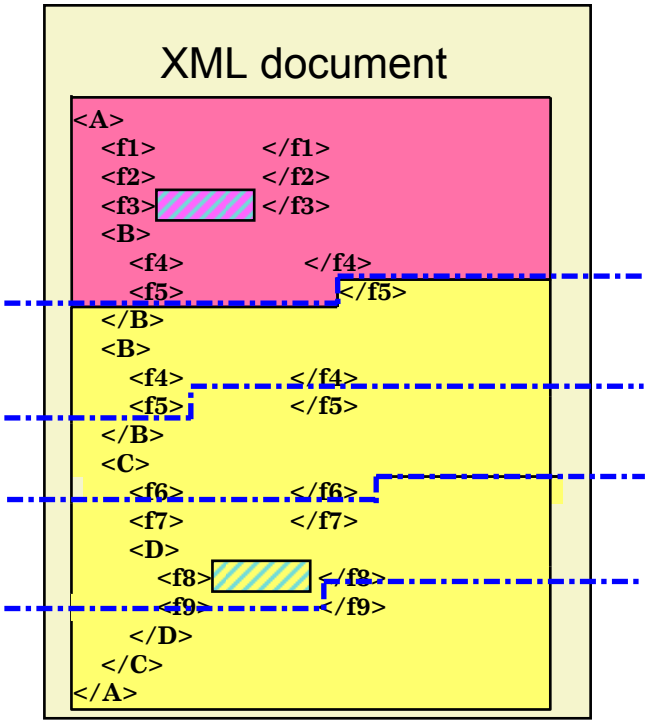
Intact Storage

- **Intact storage**
  - Insert/retrieve/delete XML documents intact in new IMS databases
    - Data stored as XML
  - Databases cannot be manipulated by non-XML applications
  - Data stored in segments based on number of bytes
    - Without regard to XML tags
  - Two segment types per document
    - One "root" and multiple overflow segments
    - Exception for secondary indexing source segments





XML schema



XML Schema

- Used to map between XML document and IMS database
  - Decomposed vs. intact
  - IMS segments containing the document
  - XML elements and attributes to/from IMS fields
- DLIModel Utility
  - Used to create XML schema
    - New function in IMS V9
    - Schemas created from PSBs, DBDs, and COBOL copybook XMI files

XML Database

- IMS Java support
  - User Defined Functions for JDBC: storeXML and retrieveXML
  - Application environments
    - IMS TM
    - CICS
    - DB2 Stored Procedures
    - WebSphere Application Server
      - Includes support for IMS Java Remote Database Services
- IMS database support
  - Full function non-HALDB
  - HALDB
  - Fast Path DEDBs

XML Database

- Benefits
  - Easy exchange of data between IMS databases and XML documents
    - Existing IMS databases may be used to create XML documents
    - Existing applications are unaffected
  - Tooling provided to define mapping
  - IMS Java support for applications

IMSplex Database Commands

- IMSplex (type-2) database commands
  - Extends IMSplex commands to IMS database resources
  - New commands
    - QUERY or QRY verb
      - DB and AREA resources
    - UPDATE or UPD verb
      - DB, AREA, and DATAGRP resources
- Benefits
  - Consistent format with IMSplex commands introduced in IMS V8
    - Especially valuable for automation
  - Consolidated responses from multiple IMS systems
  - Use of wildcards with resource names

IMS V9 has new terminology for commands:

- Type-1 commands:
  - /DIS, /START, etc.
- Type-2 commands:
  - QRY, UPD, INIT, etc.

IMSplex (Type-2) Database Commands

- Examples of QUERY or QRY:

Type-2 Command	Equivalent Type-1 Command
QUERY DB(ABC)	/DIS DB ABC
QRY DB(AB*)	No equivalent
QRY AREA NAME(*) STATUS(STOPPED)	/DIS AREA STOPPED

- Examples of UPDATE or UPD:

Type-2 Command	Equivalent Type-1 Command
UPDATE DB NAME(ABC) START(ACCESS)	/START DB ABC
UPD AREA NAME(XYZ) STOP(SCHD)	/STOP AREA XYZ
UPD DB NAME(AB*) START(ACCESS)	No equivalent

Disabling of z/OS DFSMS V1R5 PS EDI

- PS EDI
  - Physical Sequential Enhanced Data Integrity
  - PS EDI prevents concurrent opens of
    - Data sets with DSORG=PS (e.g. OSAM)
    - Allocated with DISP=SHR
    - Opened for output or update
  - New optional function in z/OS DFSMS V1R5
    - WARN mode: only send message if rule violated
    - ENFORCE mode: prevent concurrent opens for update
    - Exception data sets may be specified
      - Concurrent opens for update allowed for them
    - Authorized programs may disable function for their data sets
      - IMS uses this technique to disable PS EDI for some data sets

Disabling of z/OS DFSMS V1R5 PS EDI

- PS EDI is not required for some IMS data sets:
  - Database data sets
    - DBRC and IRLM provide data integrity
  - OLDS, WADS, RDS, and MSDB dump data sets with XRF
    - Alternate must have write capability for takeover
  - PS EDI is disabled by IMS for these data sets
    - APARs supplying this function:
      - PQ83940 for IMS V9
      - PQ83941 for IMS V8
      - PQ83942 for IMS V7
- PS EDI is excellent function for other IMS data sets
  - OLDS and WADS when not in XRF takeover

Improved Message with DB Abends

- Message identifies database involved in abend
  - Previously only issued for HALDB
    - Now issued for non-HALDB full function databases
      - Such as U085x (DFS554A is also issue
- Benefits
  - Easier and quicker identification of database abend problems
- Example

```
DFS0832I ABEND Uwwww REASON CODE xxxx yyyyyyyy
zzzzzzzz
```

`wwww` – abend code

`xxxx` – reason code

`yyyyyyyy` – ‘`PARTITION`’ for HALDB
‘`DATABASE`’ for non-HALDB

`zzzzzzzz` – partition or database name

DBRC Command Authorization for /RMxxxx

- IMS V8 added authorization for DBRC commands
  - Support for DBRC utility (DSPURX00) and HALDB Partition Definition Utility
  - Did not include /RMxxxx commands
- IMS V9 support
  - Support for /RMxxxx commands
    - RACF (or equivalent)
    - DBRC Command Authorization Exit routine (DSPDCAX0)
      - Optional
- Benefits
  - Extends consistent security to online DBRC commands

DBRC Command Authorization for /RMxxxx

- DBRC command authorization invocation
  - Same for utilities and /RMxxxx commands
  - Activated by
    - CHANGE.RECON CMDAUTH (SAF|EXIT|BOTH|NONE,safhlq)
  - Profiles can differ for different RECONs
    - Controlled by safhlq
  - Commands can be authorized at
    - Command verb level
      - For example, GENJCL command
    - Command verb + resource type level
      - For example, GENJCL.RECOV command
    - Command verb + resource type + resource name level
      - For example, GENJCL.RECOV DBD(ACCDB)

More than 32K Database Registrations

- Previous releases only allowed 32,767 registrations of databases
  - Deleting a database did not free a global DMB number in RECONs
- IMS V9 allows more than 32,767 registrations of databases
  - Maximum databases rgistered at any time is still 32,767
- Benefit
  - Avoids potetial problem for installations with many databases

DBRC – HALDB Dynamic Allocation by GENJCL.IC

- GENJCL.IC does not generate DD statements for HALDB data sets
  - Supplied ICJCL skeletal JCL member changed from IMS V8
  - Dynamic allocation is used

- Benefit
  - Especially useful for HALDB Online Reorganization users
    - IC utilities determine the active data set, then dynamically allocates it

DBRC Application Programming Interface (API)

- DBRC API
  - Allows users to write programs to read RECONs
    - Provides API to return information from all records
  - Sample program is provided

- Benefits
  - Supported method for retrieving RECON data
    - Easier than parsing LIST.RECON output
    - Easier than writing VSAM application to read RECONs
  - Release independent
    - Future releases will not require modifications to programs

DBRC Application Programming Interface (API)

- API Overview
  - Assembler language
    - Assembler macros are provided
  - Functions
    - Start the environment
      - Allocates and opens the RECONs
    - Query the RECONs
      - Acquires storage and places information in this storage
    - Release buffer storage
      - Releases storage acquired for queries
    - Stop the environment
      - Closes and deallocates RECONs

Example:

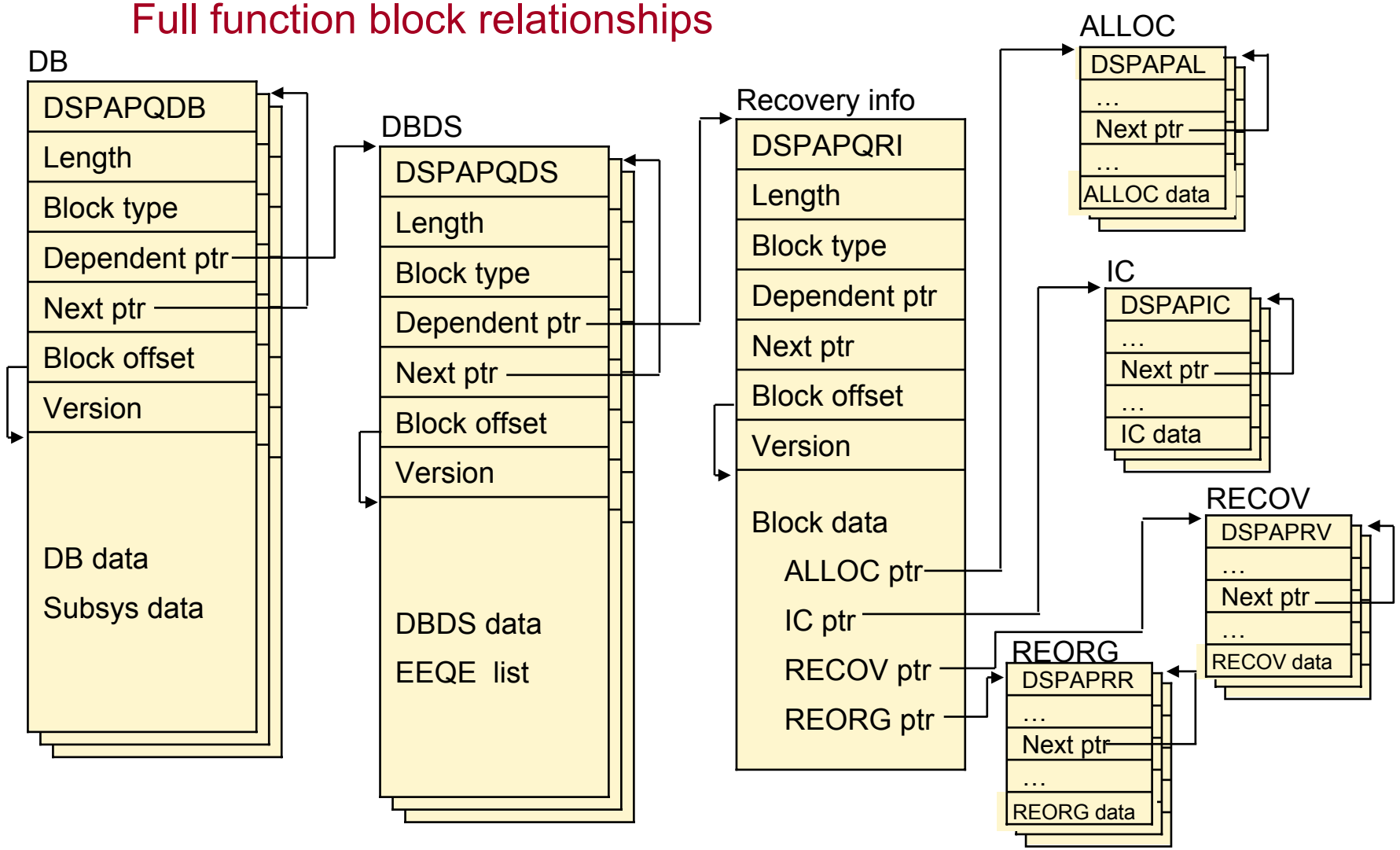
```
DSPAPI FUNC=DSECT
DSPAPQxx
DSPAPQxx

DSPAPI FUNC=STARTDBRC

DSPAPI FUNC=QUERY
Process results
DSPAPI FUNC=QUERY
Process results
DSPAPI FUNC=RELBUF

DSPAPI FUNC=QUERY
Process results
DSPAPI FUNC=RELBUF


DSPAPI FUNC=STOPDBRC
```

DBRC Application Programming Interface (API)

- Using queries
  - QUERY may ask for
    - Records by name (e.g. database name)
    - Records by list of names
    - First record (e.g. first database record)
    - Next record (e.g. next database record after specified database
    - Set of records (e.g. database record and its data set and image copy records)
  - QUERY returns blocks of data
    - Blocks are chained in RECON hierarchy
    - Blocks are mapped by DSPAPQxx macros
  - Application program follows pointers between blocks to access the data

DBRC Application Programming Interface (API)

Full function block relationships



DBRC Application Programming Interface (API)

- Summary
  - Assembler interface for users to access RECON data
  - Sample program is provided
- Benefits
  - Supported method for retrieving RECON data
  - Release independent
    - Future releases will not require modifications to programs

DB and DBRC Enhancements

- Database Enhancements
  - HALDB Online Reorganization
  - HALDB Specific Partition Initialization
  - Image Copy Large Tape Block Sizes
  - IRLM 2.2
  - Multi-Area Structures for DEDB SVSO
  - Fast Path Area Open/Close Enhancements
  - XML DB
  - IMSplex Database Commands
  - PS EDI
  - Improved Messsage with Database Abends
- DBRC Enhancements
  - Command Authorization for /RMxxxx commands
  - More than 32K Database Registrations
  - HALDB Dynamic Allocation by GENJCL.IC
  - DBRC Application Programming Interface (API)

Session Objectives

- After this session you should be able to:
  - Explain the enhancements to IMS database in V9
  - Explain the enhancements to DBRC in IMS V9

IBM Software Group

Programming to Access the RECON in IMS V9

Geoff Nicholls

Certified Senior IT Specialist

DB2 and IMS Technical conference

Mai 2005

Orlando, Florida

ON DEMAND BUSINESS

Abstract

- IMS Version 9 introduces an Application Programming Interface for access to the information in the RECON.
- This session demonstrates how to write a program to use this API for extracting information from DBRC, including setting up the environment, opening the RECON, finding the records you want, and extracting them ready for your own processing.

Agenda

- IMS Version 9 DBRC API
  - The Theory
- A Sample Assembler Program
  - The Reality
- Control Blocks returned to your program
  - The Information

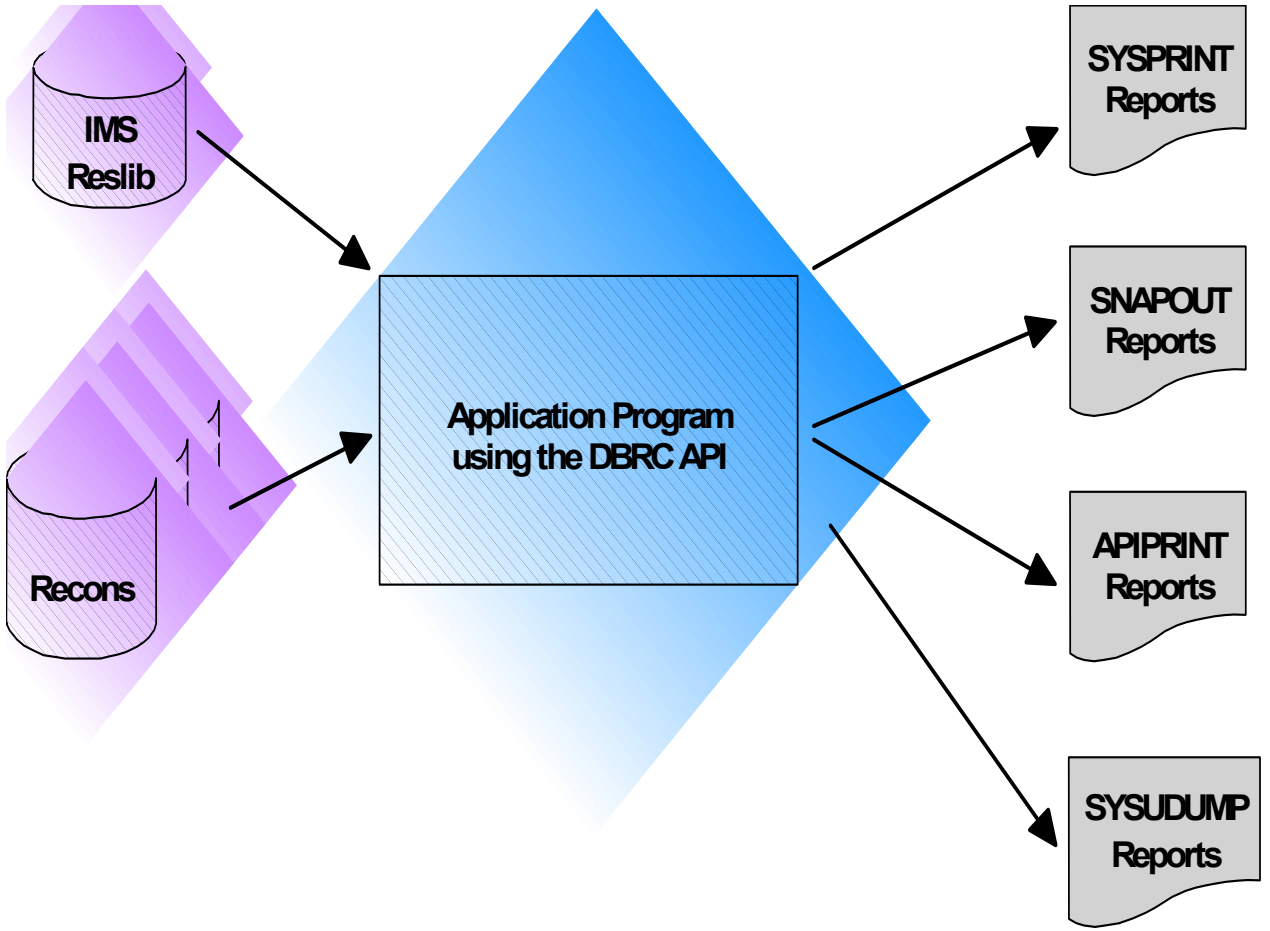
DBRC Application Programming Interface

- Requirement
  - Access the information in the RECONs
  - Provide a supported interface
- Allows users to write programs to read data in the RECONs
  - Provides API to return information from all records
  - A sample program using the API is provided
- Benefits
  - Supported method for retrieving data
    - Easier than parsing the output of LIST.RECON commands
  - Release independent
    - Future releases will not require modifications to programs
  - Users do not need to understand the segmenting of records

DBRC API Environment

- Supports only IMS Version 9 RECONs
- Assembler macros are in ADFSMAC and SDFSMAC
  - Invoke modules in SDFSRESL
- Allocates RECONs
  - Using the same mechanisms as the utilities
    - Enqueues and reserves
- IMSplex support
  - PLEX name in JCL or SCI registration exit routine
- Execution environment
  - DBRC executes in application program address space
  - AMODE 31/RMODE ANY
  - Cross memory mode is not supported
  - Calls must be made from the TCB that called STARTDBRC

Datasets Used by the Sample Program



DBRC API Functions

- Start the API environment
  - Opens the RECONs
- Query the RECONs
  - Queries for various record types
- Release buffer storage
  - Releases storage acquired for queries
- Stop the API environment
  - Deallocates the RECONs

Typical Program Structure

- Include the API DSECTs and supply working storage
  - DSPAPI FUNC=DSECT
  - DSPAPQxx macros
- Initialize the API (open RECONs and receive the API token)
  - DSPAPI FUNC=STARTDBRC
- Issue one or more Query requests
  - DSPAPI FUNC=QUERY
- Process the information returned from the Query requests
- Release buffer storage returned for each QUERY request
  - DSPAPI FUNC=RELBUF
- Terminate the DBRC API
  - DSPAPI FUNC=STOPDBRC

Application Data Definitions

- DSPAPI FUNC=DSECT
  - Includes required DSECTs and equates used by the API
- DSPAPQxx
  - Definitions of the blocks returned

Working Storage (sample from DSECT)

```
+DSPAPI_BlkType_RECON      EQU   X'0100'   RECON Status                01-DSPAP
+DSPAPI_BlkType_PRILOG     EQU   X'0500'   PRIMARY LOG                 01-DSPAP
+DSPAPI_BlkType_LOGINFO    EQU   X'0501'   Log Information             01-DSPAP
+DSPAPI_BlkType_LOGALL     EQU   X'0700'   LOG ALLOCATION              01-DSPAP
+DSPAPI_BlkType_SECLOG     EQU   X'0900'   SECONDARY LOG              01-DSPAP
+DSPAPI_BlkType_CAGROUP    EQU   X'0F00'   CA GROUP                    01-DSPAP
+DSPAPI_BlkType_CA         EQU   X'1100'   CA EXECUTION                01-DSPAP
+DSPAPI_BlkType_DBDSGRP    EQU   X'1600'   DBDS GROUP                  01-DSPAP
+DSPAPI_BlkType_DBGRP      EQU   X'1601'   DB GROUP                    01-DSPAP
+DSPAPI_BlkType_RECOVGRP   EQU   X'1602'   RECOVERY GROUP              01-DSPAP
+DSPAPI_BlkType_IMSDB      EQU   X'1800'   IMS DB                      01-DSPAP
+DSPAPI_BlkType_HALDB      EQU   X'1801'   HALDB                       01-DSPAP
+DSPAPI_BlkType_FPDEDB     EQU   X'1802'   Fast Path DEDB              01-DSPAP
+DSPAPI_BlkType_DBNotFound EQU   X'18FF'   DB Not Found                 01-DSPAP
+DSPAPI_BlkType_PART       EQU   X'1900'   PARTition                   01-DSPAP
+DSPAPI_BlkType_DBDS       EQU   X'2000'   DBDS HEADER                 01-DSPAP
+DSPAPI_BlkType_AREA       EQU   X'2100'   Area                        01-DSPAP
```

DBRC API Data Block Definitions

- DSPAPQAR Fast Path Area block
- DSPAPQBO Backout block
- DSPAPQCA CA Execution block
- DSPAPQCG CA Group block
- DSPAPQDB Full Function DB block
- DSPAPQDG DBDS, Database, and Recovery Group block
- DSPAPQDS Database Data Set (DBDS) block
- DSPAPQELEEQE List
- DSPAPQFD Fast Path DEDB block
- DSPAPQGG Global Service Group block
- DSPAPQHB HALDB block
- DSPAPQHD DBRC API Query output header
- DSPAPQHP HALDB Partition block
- DSPAPQIC Image Copy block
- DSPAPQLALog Allocation block
- DSPAPQLG RLDS/SLDS block
- DSPAPQLI Log Information block
- DSPAPQNF Not Found block
- DSPAPQOL Online Log Data Set block
- DSPAPQRC RECON Status block
- DSPAPQRI DBDS/Area Recovery Information block
- DSPAPQRR DBDS Reorganization block
- DSPAPQRV DBDS Recovery block
- DSPAPQSLDB/Area Authorized Subsystem List block
- DSPAPQSS Subsystem block

DSPAPQHD – RECON Header (sample)

```
+*****************************************************************
+DSPAPQHD             DSECT              OUTPUT HEADER
+                     DS  0F
+APQHD_EYECATCHER DS CL8              Output area eyecatcher
+APQHD_LENGTH        DS  F               Block length, hdr + data
+APQHD_BLKTYPE       DS  XL2             Block type
+                     DS  H               Reserved
+APQHD_DEPPTR        DS  A               Ptr to block dependent
+APQHD_NEXTPTR       DS  A               Ptr to next block of the same type
+APQHD_BLKOFFSET  DS F                Offset to block data
+APQHD_VERSION       DS F                Version of output block
+APQHD_APQHD_LNG    EQU *-APQHD_EYECATCHER Length of header
```

Application Processing - Initialisation

- DSPAPI FUNC=STARTDBRC
  - Returns API TOKEN
    - Used for subsequent macro calls
  - Opens RECONs
    - RECONs allocated using MDA if not previously allocated by user
      - User may allocate RECONs with DD statements or by dynamic allocation (SVC 99)
  - Opens output data set (SYSPRINT)
    - Used for any API messages
      - DD name may be overridden

Release Independence of Programs

- API macros have version parameter
  - User specifies the version
    - Version allows programs to remain the same
      - Even when new data is added by an IMS release
    - Version determines the parameters which may be specified
    - Version determines the information returned and format of information returned
    - New data is only returned when new version is specified
  - Version may change with IMS releases
    - Version for function or type will be changed only when parameters or returned data is changed
    - Version for IMS V9 is '1.0'
    - '1.0' is the default

API Token

- Use of token
  - The API token is a four-byte field
    - Used to relate a series of API requests
  - Returned by DSPAPI FUNC=STARTDBRC
  - Must be supplied on subsequent macro calls
  - No longer valid after a DSPAPI FUNC=STOPDBRC macro call

Storage Use by Programs using the API

- Queries cause IMS to acquire storage
  - DBRC never releases this storage
    - DSPAPI FUNC=STOPDBRC does not release storage
  - Application should release it with:
    - DSPAPI FUNC=RELBUF BUFFER=addr
    - BUFFER= points to first block in chain to be released
      - Chain was created by a DSPAPI FUNC=QUERY invocation

Application Processing - Query

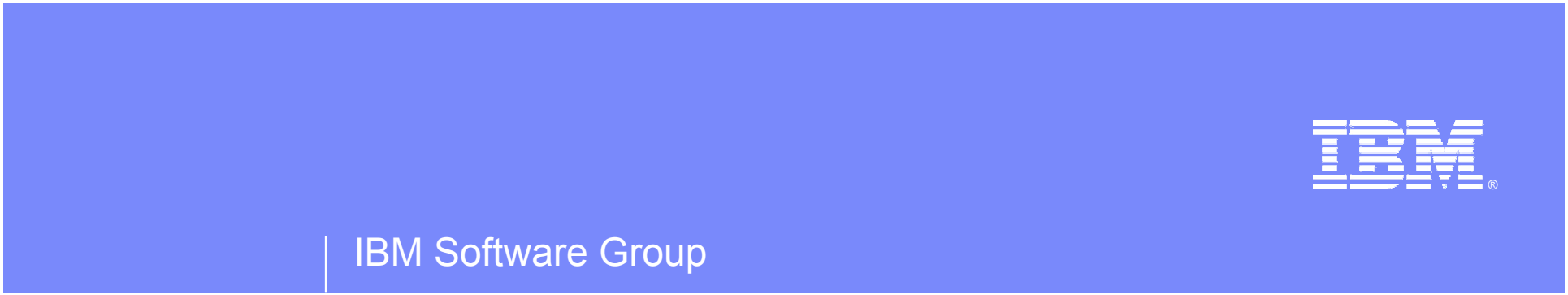
- DSPAPI FUNC=QUERY,TYPE=xxxxxxxx,...
  - xxxx specifies the type of information
    - RECON: RECON status
    - DB: Database
    - xxxGROUP: Groups
    - LOG: RLDS, SLDS
    - OLDS: OLDS
    - SUBSYS: Subsystem
    - BACKOUT: Backout
  - Acquires storage to hold information and returns the address of the storage to your program

Application Processing – Releasing Storage

- DSPAPI FUNC=RELBUF
  - Releases the storage acquired by IMS to hold the information returned in a QUERY call
    - Required to release this storage

Application Processing – Finalising Access

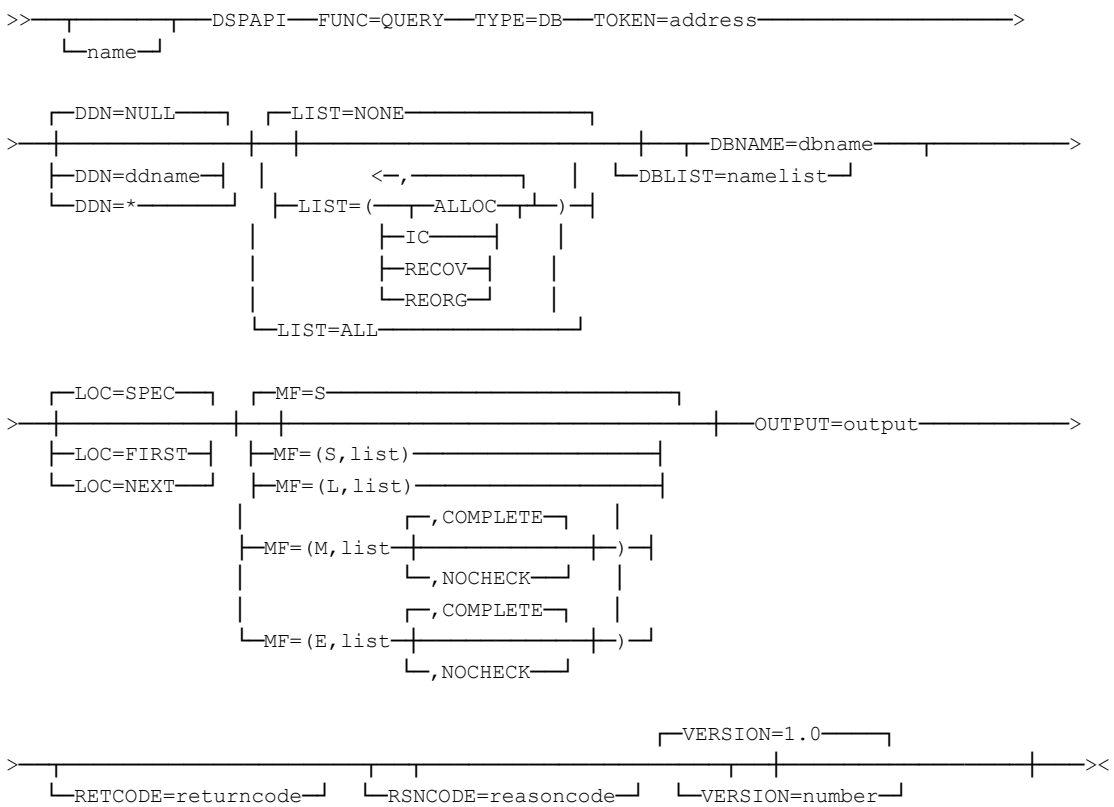
- DSPAPI FUNC=STOPDBRC
  - Closes RECONs
    - Deallocates RECONs if STARTDBRC allocated them
  - Closes SYSPRINT
  - Does not release storage holding QUERY information



Accessing Database Information from the RECON

Full Function Database Information

Application Processing - Database Query



FUNC=QUERY,TYPE=DB Examples

- DSPAPI FUNC=QUERY,TYPE=DB,LOC=FIRST,...
  - LOC=FIRST returns first database in RECONs
- DSPAPI FUNC=QUERY,TYPE=DB,DBNAME=ACCDB,...
  - DBNAME=name returns information for the named database
- DSPAPI FUNC=QUERY,
  TYPE=DB,DBNAME=ACCDB,DDN=ACCDS1,LIST=(ALLOC,IC),...
  - DDN=ACCDS1 returns information on DBDS ACCDS1
  - LIST= returns the requested recovery information
- DSPAPI FUNC=QUERY,TYPE=DB,DBNAME=ACCDB,LOC=NEXT,...
  - LOC=NEXT returns next database after the named database
- DSPAPI FUNC=QUERY,TYPE=DB,DBLIST=MYLIST,...
  - DBLIST=MYLIST returns the databases in user's list

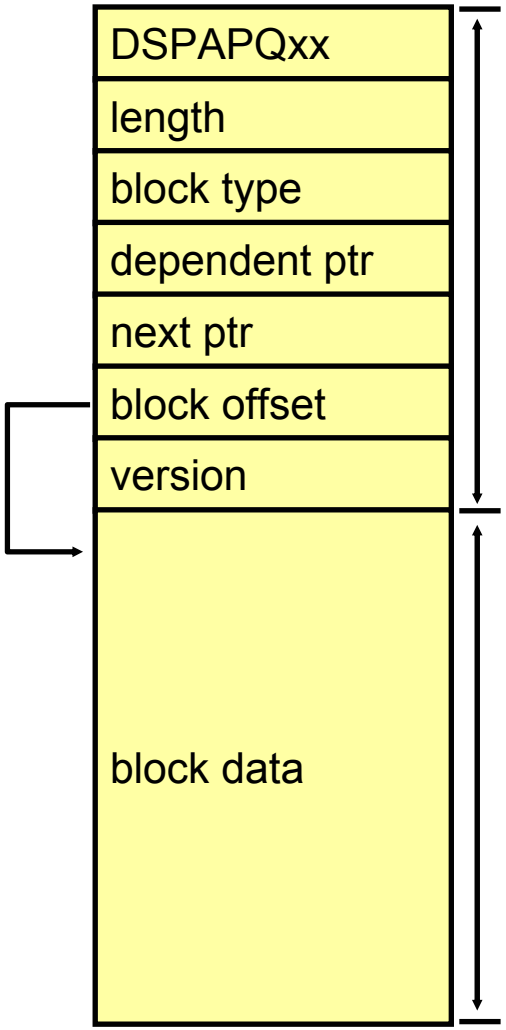
Information returned from a Query

- Returns blocks of storage with data
- Requested data is returned in blocks
  - Data is chained with pointers between blocks
- Blocks are chained in a hierarchy

Example for full function (non-HALDB) databases

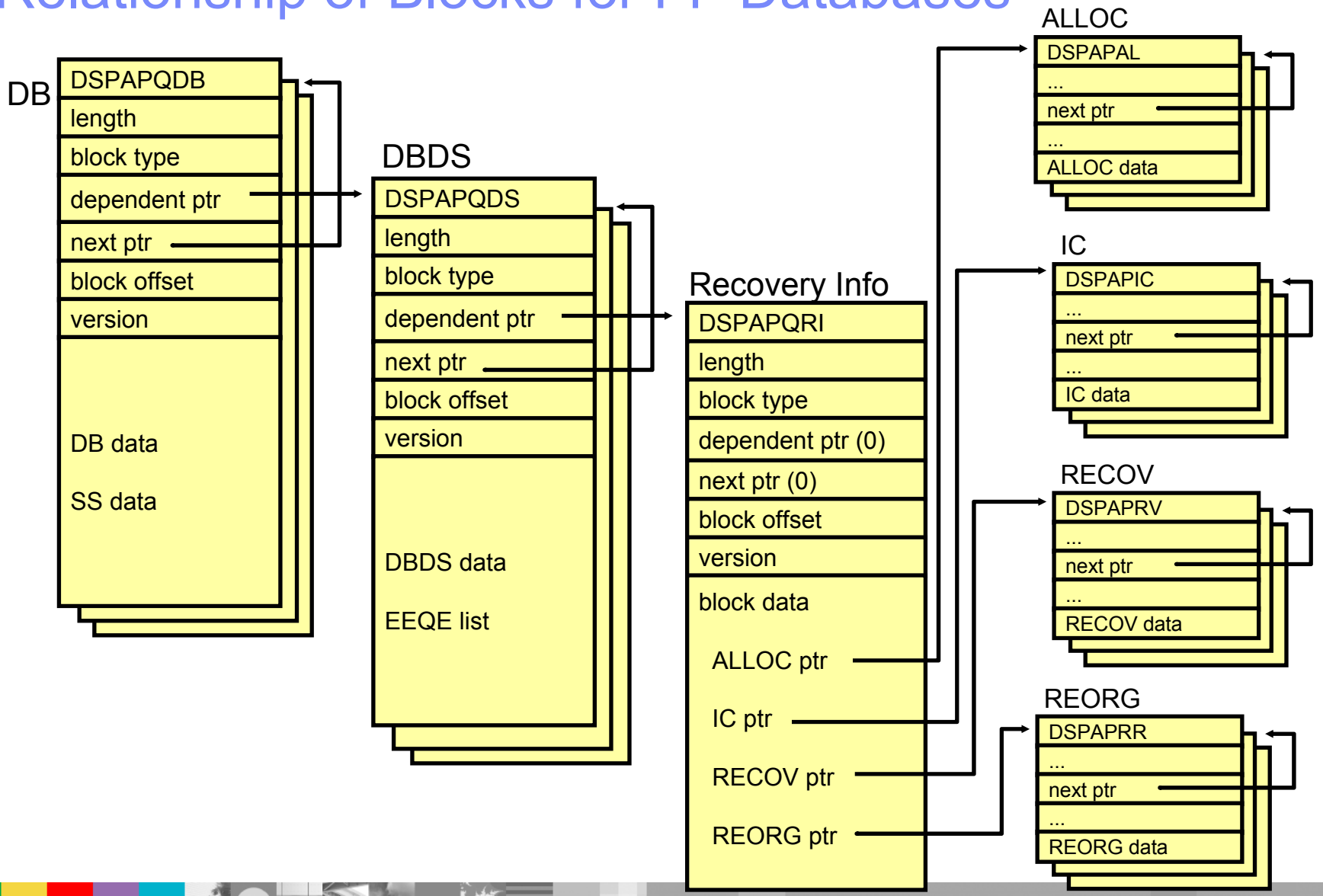
- Database block
  - Database block points to block for next database in list (if any)
  - Database block points to first DBDS block for database (if requested)
- DBDS block
  - DBDS block points to next DBDS block for database (if any)
  - DBDS block points to recovery information for the DBDS (if requested)
- Recovery information block
  - Recov info block points to Alloc, Image Copy, Recovery, and Reorg blocks
  - Alloc, Image Copy, Recovery, and Reorg blocks point to next block of the same type (Alloc to next Alloc, IC to next IC, etc.)

Format of Blocks



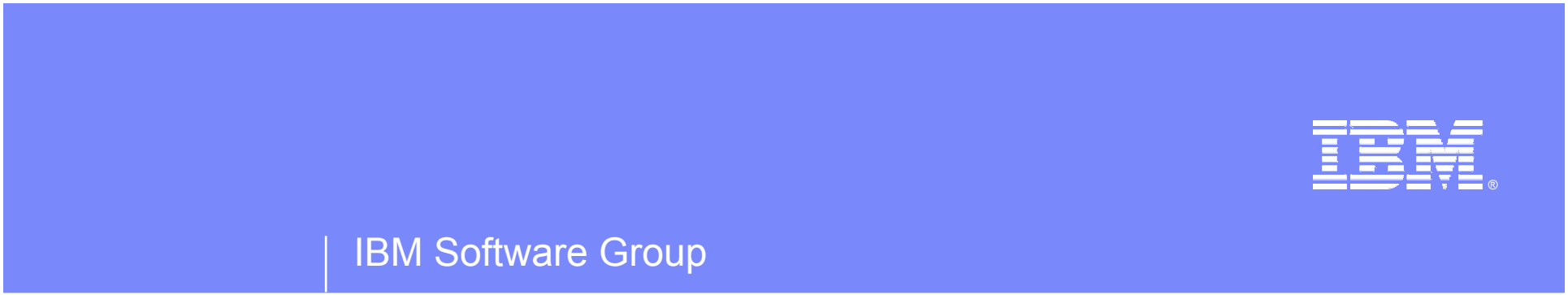
- DSPAPQxx
  - mapping macro for block data
- block type
  - IMSDB, HALDB, PART, DBDS, FPDEDB, AREA, ALLOC, REORG, CAGROUP, PRILOG, LOGALL, SUBSYS, and many more
- dependent ptr
  - pointer to a dependent block type
- next ptr
  - pointer to next block of the same type
- version
  - version of block

Relationship of Blocks for FF Databases



DBRC API Summary

- Supported method for retrieving data
  - Easier than parsing the output of LIST.RECON commands
- Consistent method for reading RECON data
  - Release independent
- Assembler macros
  - A sample program using the API is provided
    - DSPAPSMP, in the ADFSSMP library
- Documented in IMS V9 DBRC Guide and Reference



The Sample Program

Using the DBRC API

Sample Program Using The DBRC API

```
         TITLE 'Example Program Using The DBRC API'                     00010000
*********************************************************************** 00020000
*    DBRC Application Programming Interface Sample Program            * 00030000
***********************************************************@SCPYRT** 00040000
*                                                                     * 00050000
*  Licensed Materials - Property of IBM                               * 00060000
*                                                                     * 00070000
*  5655-J38                                                           * 00080000
*                                                                     * 00090000
*  (C) Copyright IBM Corp. 1974,1998 All Rights Reserved.             * 00100000
*                                                                     * 00110000
*  US Government Users Restricted Rights - Use, duplication or        * 00120000
*  disclosure restricted by GSA ADP Schedule Contract with            * 00130000
*  IBM Corp.                                                          * 00140000
***********************************************************@ECPYRT** 00150000
*                                                                     * 00160000
*  Function:                                                          * 00170000
*    This program initializes and established a DBRC Application      * 00180000
*    Programming Interface (API) environment, then makes several      * 00190000
*    calls to the API to QUERY information from DBRC.                 * 00200000
```

JCL used to execute this program

```
*    The JCL used to execute this program is:                          * 00220000
*                                                                      * 00230000
*    //DSPAPSMP JOB ACCT,NAME,MSGLEVEL=(1,1),REGION=0M                 * 00240000
*    //STEP1  EXEC  PGM=DSPAPSMP     PARM='IMSPLEX=PLEX1'              * 00250000
*    //STEPLIB  DD  DSN=SDSFRESL,DISP=SHR                              * 00260000
*    //SYSPRINT DD  SYSOUT=A                                           * 00270000
*    //APIPRINT DD  SYSOUT=A                                           * 00280000
*    //SYSUDUMP DD  SYSOUT=A                                           * 00290000
*    //SNAPOUT  DD  SYSOUT=A                                           * 00300000
*    //*RECON1  DD  DSN=RECON1,DISP=SHR                                * 00310000
*    //*RECON2  DD  DSN=RECON2,DISP=SHR                                * 00320000
*    //*RECON3  DD  DSN=RECON3,DISP=SHR                                * 00330000
*    //IMSDALIB DD  DSN=DYNALLOC,DISP=SHR                              * 00340000
*    //                                                                * 00350000
```

Get storage for the work area

```
*********************************************************************  00610000
* Get storage for the work area. See label MYDSECT at end of the     *  00620000
* listing for a mapping of the area. The storage is obtained below   *  00630000
* the line because it will contain the DCBs used by the program.     *  00640000
*********************************************************************  00650000
         STORAGE OBTAIN,LENGTH=MYDSLEN,LOC=BELOW                        00660000
         LR    R11,R1            Save the address of the storage      00670000
         USING MYDSECT,R11       Set addressability to the storage    00680000
         LR    R0,R1                                                  00690000
         LA    R1,MYDSLEN                                             00700000
         LHI   R14,0                                                  00710000
         LHI   R15,0                                                  00720000
         MVCL  R0,R14            Clear the work area storage          00730000
         MVC   SAVW2,=CL4'F1SA'  Set up my Linkage stack.             00740000
         LA    R13,SAVEAREA      Set addressability to my save area   00750000
         MVC   SNAPDCB(SNAPLEN),MYSNAP     Move in the model DCB      00760000
         MVC   PRINTDCB(PRINTLEN),MYPRINT  Ditto                      00770000
```

Write to an output data set

```
*********************************************************************  00780000
* This program writes to an output data set using a DD name of       *  00790000
* SYSPRINT and uses the MVS macro SNAPX to dump output blocks        *  00800000
* returned by the API. OPEN both the SYSPRINT and the SNAPOUT DCBs   *  00810000
* using an OPEN list in MYDSECT.                                     *  00820000
*********************************************************************  00830000
         LA    R1,MYOPEN                                               00840000
         OI    8(R1),X'80'       Indicate the end of the list          00850000
         OPEN  (SNAPDCB,OUTPUT,PRINTDCB,OUTPUT),MF=(E,(R1)),MODE=31    00860000
         LA    R1,PRINTDCB                                             00870000
         LA    R2,BGINLINE                                             00880000
* Write a message showing the program has started.                     00890000
         PUT   (R1),(R2)         Write out first line                  00900000
```

Start the API

```
*********************************************************************  00910000
* Now start the API. This macro uses the Execute form and specifies  *  00920000
* a parm list named PLIST1. The label PLIST1 is defined by a List     *  00930000
* form of the macro that is within MYDSECT.                           *  00940000
*                                                                     *  00950000
* Because SYSPRINT is used by the program, the API is told to use     *  00960000
* a different output data set using the DD name APIPRINT.             *  00970000
*********************************************************************  00980000
* Issue FUNC=STARTDBRC                                                  00990000
         DSPAPI  FUNC=STARTDBRC,TOKEN=MYTOKEN,SYSPRINT=ALTPRINT,      C01000000
               MF=(E,PLIST1,NOCHECK)                                    01010000
         STM     R15,R0,MYRETCD                                         01020000
         LA      R1,PLIST1                                              01030000
* Write a line showing the results of the call.                         01040000
         BAS     R14,Print_Result                                       01050000
         L       R7,MYRETCD                                             01060000
         LTR     R7,R7                                                  01070000
*********************************************************************  01080000
* If FUNC=STARTDBRC failed, just exit. No need to do any more.       *  01090000
*********************************************************************  01100000
         JNZ     GETOUT                                                 01110000
```

QUERY the RECON records

```
*********************************************************************  01240000
* OK, FUNC=STARTDBRC worked, QUERY the RECON record(s). This will    *  01250000
* work because FUNC=STARTDBRC would have failed if there were no     *  01260000
* RECONs to work with.                                               *  01270000
*********************************************************************  01280000
* Issue FUNC=QUERY for RECON                                           01290000
         DSPAPI FUNC=QUERY,TYPE=RECON,                               C01300000
               TOKEN=MYTOKEN,                                        C01310000
               OUTPUT=MYOUTPT,                                       C01320000
               MF=(E,PLIST2,COMPLETE)                                 01330000
         STM   R15,R0,MYRETCD                                         01340000
         LA    R1,PLIST2                                              01350000
         BAS   R14,Print_Result                                       01360000
         L     R7,MYRETCD                                             01370000
* Something is seriously wrong if the call failed.                    01380000
         LTR   R7,R7                                                  01390000
         JNZ   STOPIT                                                 01400000
*********************************************************************  01410000
* QUERY worked, use SNAPX to dump the output block(s).               *  01420000
*********************************************************************  01430000
         LA    R9,MYSAVE                                              01440000
         L     R8,MYOUTPT                                             01450000
         BAS   R14,SNAPX_Blocks                                       01460000
```

Free the blocks returned

```
*********************************************************************  01470000
* Now free the block(s) returned by the QUERY call.                 *  01480000
*********************************************************************  01490000
* Issue FUNC=RELBUF                                                    01500000
         DSPAPI  FUNC=RELBUF,TOKEN=MYTOKEN,BUFFER=MYOUTPT,           C01510000
               VERSION=1.0,                                          C01520000
               MF=(E,PLIST1,COMPLETE)                                 01530000
         STM    R15,R0,MYRETCD                                        01540000
         LA     R1,PLIST1                                             01550000
         BAS    R14,Print_Result                                      01560000
*********************************************************************  01570000
* Notice that no field is specified on the DSPAPI macros for the    *  01580000
* return code and reason code. Instead, the program stores them from *  01590000
* R15 and R0. This is because the API attempts to validate any input *  01600000
* addresses and to recover from an ABEND. If the RETCODE or RSNCODE  *  01610000
* field failed validation, it would be a program error to check      *  01620000
* those fields. In this call, storage location 1 is passed as the   *  01630000
* location to receive the return code. The call will fail and an    *  01640000
* ABEND dump will be produced. It will be ignored.                  *  01650000
*********************************************************************  01660000
```

Locate the first DB record

```
*********************************************************************  01760000
* Locate the first DB record. This call may fail if there are no     *  01770000
* DB records in the RECON.                                           *  01780000
*********************************************************************  01790000
* Issue FUNC=QUERY for first DB                                        01800000
         LA    R7,MYOUTPT                                             01810000
         DSPAPI FUNC=QUERY,TYPE=DB,LOC=FIRST,                        C01820000
               TOKEN=MYTOKEN,                                        C01830000
               OUTPUT=(R7),DDN=NULL,LIST=NONE,                       C01840000
               MF=(E,PLIST2,COMPLETE)                                 01850000
         STM   R15,R0,MYRETCD                                         01860000
         LA    R1,PLIST2                                              01870000
         BAS   R14,Print_Result                                       01880000
         L     R7,MYRETCD                                             01890000
         LTR   R7,R7                                                  01900000
*********************************************************************  01910000
* If FUNC=QUERY failed, try to locate a log record.                  *  01920000
*********************************************************************  01930000
         JNZ   QRYLOG                                                 01940000
*********************************************************************  01950000
* Use SNAPX to dump the output block(s).                             *  01960000
*********************************************************************  01970000
         LA    R9,MYSAVE                                              01980000
         L     R8,MYOUTPT                                             01990000
         BAS   R14,SNAPX_Blocks                                       02000000
```

Query a Database List

```
*********************************************************************  02010000
* Using the DB name returned by the QUERY FIRST, QUERY a DBLIST.     *  02020000
* The first DB in the list will fail because the name is invalid.    *  02030000
* The second will work because it is the name returned from above.   *  02040000
*********************************************************************  02050000
         L     R7,MYOUTPT                  Point at the HDR             02060000
APQHD2   USING DSPAPQHD,R7                                              02070000
         A     R7,APQHD2.APQHD_BLKOFFSET   Increment to DB info         02080000
         DROP  APQHD2                                                   02090000
APQDB1   USING DSPAPQDB,R7                                              02100000
         MVC   DBLST2,APQDB1.APQDB_DBNAME  Save the DB name             02110000
         DROP  APQDB1                                                   02120000
         LA    R7,2                                                     02130000
         ST    R7,DBLSTCNT                                              02140000
         SR    R7,R7                                                    02150000
         ST    R7,DBLST1                                                02160000
```

Close the DCBs and Free any storage obtained

```
*********************************************************************  02920000
* All done. CLOSE the DCBs and FREE any storage obtained by this     *  02930000
* program.                                                           *  02940000
*********************************************************************  02950000
*                                                                       02960000
GETOUT   LA    R1,PRINTDCB                                             02970000
         LA    R2,ENDLINE                                              02980000
         PUT   (R1),(R2)         Write out first line                  02990000
         LA    R1,MYCLOS                                               03000000
         OI    8(R1),X'80'         Indicate end of the CLOSE list      03010000
         CLOSE (SNAPDCB,,PRINTDCB),MF=(E,(R1)),MODE=31                 03020000
* FREE my storage                                                       03030000
GETOUT2  STORAGE RELEASE,ADDR=(R11),LENGTH=MYDSLEN                      03040000
MYEXIT   SR    R15,R15                                                 03050000
         PR                        Return to the system                03060000
         EJECT                                                         03070000
```

DBRC API Storage Definition Macros

```
*********************************************************************** 06070000
* DSPAPI FUNC=DSECT includes constants and equates needed by the API * 06080000
*********************************************************************** 06090000
         DSPAPI FUNC=DSECT                                              06100000
*********************************************************************** 06110000
* DSPAPQHD maps the header block for all output returned by the API  * 06120000
*********************************************************************** 06130000
         DSPAPQHD                                                       06140000
*********************************************************************** 06150000
* DSPAPQDB maps the DB output block.                                 * 06160000
*********************************************************************** 06170000
         DSPAPQDB                                                       06180000
*********************************************************************** 06190000
* DSPAPQLI maps the LOGINFO output block.                            * 06200000
*********************************************************************** 06210000
         DSPAPQLI                                                       06220000
*********************************************************************** 06230000
* DSPAPQRI maps the RCVINFO output block.                            * 06240000
*********************************************************************** 06250000
         DSPAPQRI                                                       06260000
         END   DSPAPSMP                                                 06270000
```

Information Returned by the API

The Blocks of Data

DSPAPQRC – RECON Status Header

```
                                                          |eyecatcher     |
21989420                                                  C4E2D7C1 D7D8D9C3   *                       DSPAPQRC*
         |BlkLen| |TP|---- |Ptr 1 | |Ptr 2 |    |Offset| |Vers'n| |Data
21989440 000002EF 01000000 00000000 00000000    00000020 00000100 D9C5C3D6 E5C5D9E8   *.....................RECOVERY*

21989460 40C3D6D5 E3D9D6D3 40C4C1E3 C140E2C5    E36B40C9 D4E240E5 F9D9F140 40404040   * CONTROL DATA SET, IMS V9R1     *
               | |Ptr    | -------- --------    |Ln||Ct|
21989480 40404040 00000230 00000000 00000000    00350300 40800000 00000000 00000000   *    ..............  ...........*
219894A0 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
      LINES 219894C0-21989500  SAME AS ABOVE
21989520 80270000 04148F09 24374F00 00000000    00000071 00000000 C9D4F1C2 40404040   *.........|.............IM1B    *
21989540 F3F3F9F0 40404040 F3F4F8F0 40404040    0000D6D5 D7F20100 00010000 001F0000   *3390    3480    ..ONP2..........*
21989560 00000000 000C0000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
21989580 00000000 00000000 00000000 00000000    00000000 00000000 00000000 00000000   *................................*
      LINES 219895A0-21989640  SAME AS ABOVE
21989660 00000000 00000000 00000000 00000000    00000000 0000000F 00000010 0000005F   *.............................¬*
21989680 00000003 00000010 D9C5C3D6 D5F14040    C9D4E2D7 E2C14BC9 D4F0C24B D9C5C3D6   *........RECON1  IMSPSA.IM0B.RECO*
219896A0 D5F14040 40404040 40404040 40404040    40404040 40404040 40404040 80D9C5C3   *N1                            .REC*
219896C0 D6D5F240 40C9D4E2 D7E2C14B C9D4F0C2    4BD9C5C3 D6D5F240 40404040 40404040   *ON2  IMSPSA.IM0B.RECON2         *
219896E0 40404040 40404040 40404040 40404040    4040D9C5 C3D6D5F3 40404040 40404040   *                  RECON3        *
21989700 40404040 40404040 40404040 40404040    40404040 40404040 40404040 40404040   *                                *
21989720 40404040 40400800                                                            *      ..                        *
```

DSPAPQHD – HALDB Database Master

```
21902120                   C4E2D7C1 D7D8C8C2    0000005C 18010000 21902C30 00000000   *        DSPAPQHB...*............*
21902140 00000020 00000100 C3E4E2E3 C4C24040    00000000 00000000 00000000 00000000   *........CUSTDB  ................*
21902160 20300301 80260003 00030003 00000000    00000000 00000000 00000000 00000000   *................................*
21902180 00000000                                                                     *....                           *
```

DSPAPQHP – HALDB Partition

```
21902C20                                          C4E2D7C1 D7D8C8D7 0000010C 19000000   *               DSPAPQHP........*
21902C40 00000000 21902EF0 00000020 00000100      C3E4E2E3 C4C24040 C3E4E2E3 C4F14040   *.......0........CUSTDB  CUSTD1  *
21902C60 000000C2 000000E2 000000C0 00000000      C9D4E2D7 E2C14BC9 D4F0C24B C3E4E2E3   *...B...S...{....IMSPSA.IM0B.CUST*
21902C80 C4C24040 40404040 40404040 40404040      40404040 40000000 00000000 D9C1D5C4   *DB                    .......RAND*
21902CA0 C3E4E2E3 0000D674 FFFFFFFF 000B0A14      0001000A 0002010C 00000000 00000000   *CUST..O........................*
21902CC0 C3E4E2E3 C4F24040 00000000 00000000      00000000 00000000 00000000 03038026   *CUSTD2  .......................*
21902CE0 00020010 00000000 00000000 00000000      00000000 00000006 00000006 00000006   *................................*
21902D00 00000006 00000000 00000001 00000000      2000C9D4 F2C24040 40400303 00000000   *................IM2B    ......*
21902D20 0000C9D4 F1C24040 40400303 00000000      0000F0F0 F1F8F9F9 F9F9F9F9            *..IM1B    ........0018999999    *
```

DSPAPQLI – Log Information

```
021902120                                                 C4E2D7C1 D7D8D3C9   *                      DSPAPQLI*
 21902140 00000050 05010000 00000000 00000000    00000020 00000100 C9D4F1C2 40404040   *...&.....................IM1B    *
 21902160 2004149F 15334580 0000016D 21902ED8    21989388 00000000 21989260 00000000   *...........3.-...Q.qlh.....qk-....*
 21902180 00000000 00000000                                                            *........                      *
```

DSPAPQLG – Log Record

```
21902EC0                                                  C4E2D7C1 D7D8D3C7   *                       DSPAPQLG*
21902EE0 00000128 05000000 00000000 00000000    00000020 00000100 00000060 00000060   *............................-...-*
21902F00 00000000 00000000 C9D4F1C2 40404040    2004149F 15334580 0000016D 2004149F   *........IM1B    ...........____....*
21902F20 15383142 0766016D 00000001 1EC00000    00000000 00000001 00000000 00000000   *.......______.....{..................*
21902F40 00000000 2004149F 15334846 0984016D    00000000 00000000 00000000 00000000   *.............d._.................*
21902F60 00000078 C9D4E2D7 E2C14BE2 D3C4E2D7    4BC9D4F1 C24BC4F0 F4F1F4F9 4BE3F1F1   *....IMSPSA.SLDSP.IM1B.D04149.T11*
21902F80 F3F3F4F5 F84BE5F0 F0404040 40404040    2004149F 15334580 0000016D 2004149F   *33458.V00       ...........____....*
21902FA0 15383142 0766016D 00000000 00000000    00000001 00000000 0000024B 00000009   *.......______.........................*
21902FC0 F3F3F9F0 40404040 00010001 00000000    00000000 E3D6E3C9 D4E50200 2004149F   *3390    ............TOTIMV......*
21902FE0 15383142 0766016D 2004149F 15383039    3149016D BB48E2D9 F7AA0000 00000000   *.......___________..SR7.......*
```

Summary

- The DBRC API is provided by IMS Version 9
- Accessible from Assembler
- Control Blocks returned